PATENT ABSTRACTS OF JAPAN

#2

(11)Publication number : 2001-230770
(43)Date of publication of application : 24.08.2001

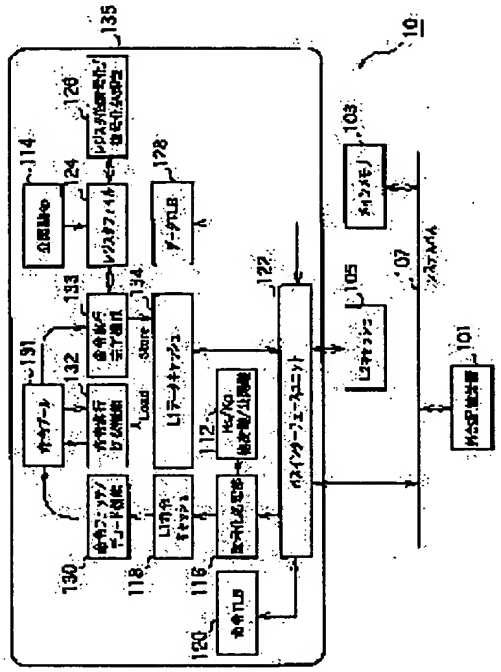
(51)Int.Cl. H04L 9/10
G09C 1/00

(21)Application number : 2000-035898
(22)Date of filing : 14.02.2000

(71)Applicant : TOSHIBA CORP
(72)Inventor : HASHIMOTO MIKIO
SAITO TAKESHI
TERAMOTO KEIICHI

(54) MICROPROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a microprocessor which can securely prevent the illegal rewriting of an operation, program.
SOLUTION: The microprocessor holds inside a specific secret key which cannot be read out and decodes contents are previously ciphered by a specific open key corresponding to the secret key. The bus interface unit of the microprocessor has an address conversion means converting a virtual address into a physical address and a memory read means reading contents corresponding to the physical address designated from a memory outside the microprocessor. Instruction TLB stores a table including an address conversion rule and one or above entries having ciphered attribute information in ranges designated by the virtual addresses. When ciphered attribute information shows that the range designated by the virtual address is ciphered, the corresponding contents are read from the outer memory through the memory reading means and it is decoded by a decoding processing section.



LEGAL STATUS

[Date of request for examination] 11.12.2002
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

MicroPatent® Worldwide PatSearch: Record 1 of 1

Family of JP2000035898 How It Works

MicroPatent® Family Lookup

Stage 2 Patent Family - "Extended"					Priorities and Applications			
	CC	Document Number	KD	Publication Date	CC	Application or Priority Number	KD	Application or Priority Date
☐	JP	2000035898	A	20000202	JP	20262598	A	19980717
☐	JP	3189793	B2	20010716	JP	20262598	A	19980717

2 Publications found.
Information on the left side of the table relates to publication number, kind, and date; information on the right covers the corresponding application and priority data for each publication.
Legend: CC=Country Code KD=Kind (Publication kind can differ from application/priority kind.)

Order Selected Documents

For further information, please contact:
Technical Support | Billing | Sales | General Information

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－35898
（P2000－35898A）
(43)公開日　平成12年２月２日(2000.2.2)

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　11/22	３４０	Ｇ０６Ｆ　11/22	３４０Ａ　５Ｂ０４８

審査請求　有　請求項の数41　ＯＬ　（全 47 頁）

(21)出願番号　特願平10－202625

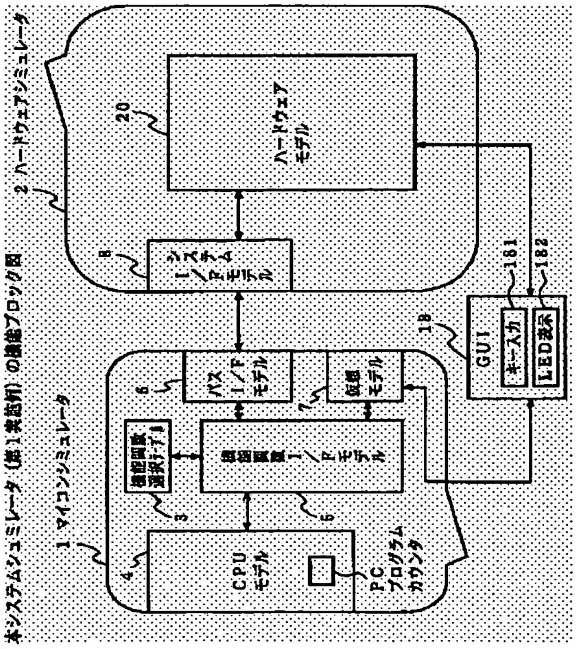
(22)出願日　平成10年７月17日(1998.7.17)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　福島　裕
東京都港区芝五丁目７番１号　日本電気株式会社内
(74)代理人　100086645
弁理士　岩佐　義幸

Ｆターム(参考)　5B048 AA12 AA20 DD14

(54)【発明の名称】　システムシミュレータおよびシステムシミュレーション方法

(57)【要約】
【課題】　マイコンを使用する電子機器のシステムシミュレーションにおいて、検証時間の短縮と検証内容の充実を図る。
【解決手段】　ハードウェアシミュレータ２は、電子機器の周辺回路をソフトウェアで模擬するハードウェアモデル２０と、マイコンシミュレータ１との間で情報授受の整合をとるシステムＩ／Ｆモデル８とから成り、電子機器のハードウェアをソフトウェアで検証する。マイコンシミュレータ１のＣＰＵモデル１はターゲットプログラムを検証する。このとき、ハードウェア機能関数Ｉ／Ｆモデル５を介して、仮想モデル７に登録されたハードウェアシミュレーションの結果を適時に利用する。機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４２からの機能関数のうち機能関数選択テーブル３に登録されているものについては仮想モデル７を利用し、そうでないものについてはバスＩ／Ｆモデル６により情報授受の整合をとってハードウェアシミュレータ２を使用する。



【特許請求の範囲】

【請求項１】マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレータにおいて、

前記プログラムに基づいて前記ハードウェアをソフトウェアで検証するハードウェアシミュレータと、

前記ハードウェアに係わる前記プログラム命令を前記ハードウェアと等価的にソフトウェアで処理する仮想モデルシミュレータと、

前記ハードウェアシミュレータまたは前記仮想モデルシミュレータの出力を適時に利用しながら前記プログラムをソフトウェアで検証するＣＰＵモデルシミュレータとを具備することを特徴とするシステムシミュレータ。

【請求項２】前記ＣＰＵモデルシミュレータは、ＣＰＵインタフェースモデルを介して前記ハードウェアシミュレータまたは前記仮想モデルシミュレータのいずれかと接続されることを特徴とする請求項１記載のシステムシミュレータ。

【請求項３】前記ＣＰＵモデルシミュレータは、前記電子機器を構成する周辺回路に対する入出力命令を機能関数に変換して出力する手段を設けたことを特徴とする請求項１または請求項２に記載のシステムシミュレータ。

【請求項４】前記ＣＰＵインタフェースモデルは、前記ハードウェアシミュレータまたは前記仮想モデルシミュレータの一方を選択して前記ＣＰＵモデルシミュレータと接続する機能関数インタフェースモデルと、

前記仮想モデルシミュレータを選択するための条件を規定したシミュレータ選択テーブルとを有することを特徴とする請求項１～請求項３のいずれかに記載のシステムシミュレータ。

【請求項５】前記ハードウェアシミュレータは、前記電子機器を構成する周辺回路対応のサブモデルを含み前記ハードウェアの検証を実行するハードウェアモデルと、該ハードウェアシミュレータと前記ＣＰＵモデルシミュレータとの間の情報授受の整合をとるシステムインタフェースモデルとを有することを特徴とする請求項１～請求項３のいずれかに記載のシステムシミュレータ。

【請求項６】前記ＣＰＵインタフェースモデルは、前記機能関数インタフェースモデルと前記システムインタフェースモデルとの間の整合をとるバスインタフェースモデルを有することを特徴とする請求項５に記載のシステムシミュレータ。

【請求項７】前記電子機器を構成する周辺回路対応のサブモデルは、前記周辺回路の接続情報を有していることを特徴とする請求項５に記載のシステムシミュレータ。

【請求項８】前記電子機器を構成する周辺回路対応のサブモデルは、前記プログラム・コード、または前記プログラムの処理結果、またはメモリモデルの記憶情報を保持する記憶手段を有していることを特徴とする請求項５に記載のシステムシミュレータ。

【請求項９】前記仮想モデルシミュレータは、前記電子機器を構成する周辺回路に対応する命令の実行結果と等価な機能を実行する複数の仮想サブモデルを有することを特徴とする請求項１～請求項３のいずれかに記載のシステムシミュレータ。

【請求項１０】前記仮想サブモデルは、前記プログラム・コード、または前記プログラムの処理結果、またはメモリモデルの記憶情報を保持する記憶手段を有することを特徴とする請求項９に記載のシステムシミュレータ。

【請求項１１】前記仮想サブモデルは、複数の前記サブモデルに係わる前記プログラムを構成するサブルーチンの機能を代行する手段で構成されることを特徴とする請求項９または請求項１０に記載のシステムシミュレータ。

【請求項１２】前記仮想サブモデルは、前記サブモデルに係わる前記プログラムを前記サブモデルと等価な処理をする周辺装置で処理する手段で構成されることを特徴とする請求項９または請求項１０に記載のシステムシミュレータ。

【請求項１３】前記仮想モデルシミュレータは、シミュレーション結果に基づき、該当する前記サブモデルの記憶手段の記憶内容を更新することを特徴とする請求項９～請求項１２のいずれかに記載のシステムシミュレータ。

【請求項１４】前記ハードウェアモデルシミュレータは、シミュレーション結果に基づき、該当する前記仮想サブモデルの記憶手段の記憶内容を更新する手段を有することを特徴とする請求項９～請求項１２のいずれかに記載のシステムシミュレータ。

【請求項１５】前記ハードウェアシミュレータは、前記ハードウェアモデルからの割込み要求信号を前記ＣＰＵモデルに伝える割込インタフェースモデルを有し、前記ハードウェアの検証を前記プログラムの検証より優先して行うようにしたことを特徴とする請求項１～請求項１４のいずれかに記載のシステムシミュレータ。

【請求項１６】前記ハードウェアシミュレータ、前記ハードウェアモデルからのバス解放要求信号を前記ＣＰＵモデルに伝え、前記ＣＰＵモデルからのバス使用許可信号を前記ハードウェアモデルに伝えるバス要求インタフェースモデルを有し、前記プログラムの検証を中断するようにしたことを特徴とする請求項１～請求項１５のいずれかに記載のシステムシミュレータ。

【請求項１７】前記ハードウェアモデルと前記仮想モデルとの対応は、各モデルを構成するサブモデル間で１：Ｎ（Ｎは２以上の整数）であることを特徴とする請求項１～請求項１６のいずれかに記載のシステムシミュレータ。

【請求項１８】前記ハードウェアモデルまたは前記仮想モデルは、各モデルに所定のデータを与えたり、シミュ

レーション結果を表示するユーザーインタフェースと接続されたことを特徴とする請求項１～請求項１７のいずれかに記載のシステムシミュレータ。

【請求項１９】マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

前記ハードウェアの接続情報に基づいて周辺回路の処理をシミュレーションする第１ステップと、

前記ハードウェアの論理機能に基づいて周辺回路の処理を等価的にシミュレーションする第２ステップとを有し、

前記第１または第２ステップのいずれかを選択してマイクロコンピュータの入出力命令をシミュレーションする第３ステップとを備えたことを特徴とするシステムシミュレーション方法。

【請求項２０】マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

前記周辺回路の接続情報に基づいて周辺回路の処理をシミュレーションする第１ステップと、

前記周辺回路の機能と等価な機能を有する周辺装置を使用して周辺回路の処理結果を得る第２ステップとを有し、

前記第１または第２ステップのいずれかを選択してマイクロコンピュータの入出力命令をシミュレーションする第３ステップとを備えたことを特徴とするシステムシミュレーション方法。

【請求項２１】前記第１または第２ステップのいずれかを選択する前記第３ステップは、シミュレーション選択テーブルに基づいて行われ、前記シミュレーション選択テーブルの選択条件を満たすとき第２ステップを選択し、満たさないとき前記第１ステップを選択するようにしたことを特徴とする請求項１９または請求項２０に記載のシステムシミュレーション方法。

【請求項２２】前記第１または第２ステップのいずれかを選択する前記第３ステップは、シミュレーション選択テーブルに基づいて行われ、

前記シミュレーション選択テーブルにフラグが付加され、前記フラグが第１の状態であって前記選択テーブルの選択条件を満たすとき前記第２ステップを選択し、前記フラグが第２の状態であるとき前記第１ステップを選択することを特徴とする請求項１９または請求項２０に記載のシステムシミュレーション方法。

【請求項２３】前記第１及び第２ステップの処理結果を第１及び第２の記憶領域にそれぞれ記憶するステップを有し、

第１ステップを実行した後に、第１ステップの処理結果で第２の記憶領域を更新するステップと、

第２ステップを実行した後に、第２ステップの処理結果で第１の記憶領域を更新するステップとを備えたことを特徴とする請求項１９または請求項２０に記載のシステムシミュレーション方法。

【請求項２４】マイクロコンピュータとこれに接続される周辺回路とを備えた電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

前記マイクロコンピュータが前記周辺回路との入出力命令を検出するステップと、

前記入出力命令に対応する機能関数を生成するステップと、

前記機能関数に応答し、前記ハードウェアの接続情報に基づいて前記周辺回路の処理をシミュレーションするハードウェアモデルシミュレーションと、

前記ハードウェアの論理機能に基づいて前記周辺回路の処理を等価的にシミュレーションする仮想モデルシミュレーションとを有し、

前記機能関数に基づき前記ハードウェアモデルシミュレーションまたは前記仮想モデルシミュレーションのいずれかを選択して前記入出力命令に対するシミュレーション結果を求めるステップを備えたことを特徴とするシステムシミュレーション方法。

【請求項２５】前記ハードウェアモデルシミュレーションまたは前記仮想モデルシミュレーションを選択する処理は、前記仮想モデルシミュレーションで処理可能な前記機能関数名と、前記機能関数の選択条件が登録されたシミュレーション選択テーブルにより行われることを特徴とする請求項２４に記載のシステムシミュレーション方法。

【請求項２６】前記ハードウェアモデルシミュレーションまたは前記仮想モデルシミュレーションを選択する処理は、前記ハードウェアモデルシミュレーションで処理可能な前記機能関数名と、前記機能関数の選択条件と、フラグとが登録されたシミュレーション選択テーブルにより行われることを特徴とする請求項２４に記載のシステムシミュレーション方法。

【請求項２７】前記仮想モデルシミュレーションへ前記ハードウェアモデルシミュレーションの初期状態と等価な状態を登録する処理はシステムシミュレーション開始前に行われていることを特徴とする請求項２４～請求項２６のいずれかに記載のシステムシミュレーション方法。

【請求項２８】前記仮想モデルシミュレーションへ前記ハードウェアモデルの検証結果を登録する処理は前記システムシミュレーションの進行につれて更新されることを特徴とする請求項２４～請求項２７のいずれかに記載

のシステムシミュレーション方法。

【請求項２９】前記更新は、前記仮想モデルに付属して設けられたテーブルまたはレジスタへの書き込みによって行うことを特徴とする請求項２８に記載のシステムシミュレーション方法。

【請求項３０】前記システムシミュレーションの進行につれて前記仮想モデルシミュレーションの結果が登録され、その登録内容が前記ハードウェアモデルシミュレーションに反映されることを特徴とする請求項２４～請求項２８のいずれかに記載のシステムシミュレーション方法。

【請求項３１】前記更新は、前記ハードウェアモデルに付属して設けられたテーブルまたはレジスタへの書き込みによって行うことを特徴とする請求項３０に記載のシステムシミュレーション方法。

【請求項３２】前記周辺回路の検証は、前記マイクロコンピュータのシミュレーションにおける周辺回路処理サブルーチンとして行われることを特徴とする請求項２４に記載のシステムシミュレーション方法。

【請求項３３】前記ハードウェアモデルシミュレションの結果をもとに割込み要求を検出するステップと、
割込インタフェースモデルを介して前記マイクロコンピュータのモデルに割込み要求信号を伝えるステップと、
前記ターゲットプログラムに記述され、前記割込み要求信号によって起動される割込み処理によりハードウェアの検証を行うようにしたことを特徴とする請求項２４～請求項３２のいずれかに記載のシステムシミュレーション方法。

【請求項３４】前記ハードウェアモデルシミュレーションの結果をもとにバス解放要求を検出するステップと、
バス解放要求インタフェースモデルを介して前記マイクロコンピュータのモデルにバス解放要求信号を伝えるステップと、
バス解放要求インタフェースモデルを介して前記ハードウェアモデルにバス使用許可信号を伝えるステップと、
前記マイクロコンピュータモデルの検証を停止させて前記ハードウェアの検証を行うようにしたことを特徴とする請求項２４～請求項３２のいずれかに記載のシステムシミュレーション方法。

【請求項３５】前記割込み要求信号は前記周辺回路を構成するシリアル入出力インタフェース対応のサブモデルから出力され、また前記バス要求信号は前記周辺回路を構成するＤＭＡ制御装置対応のサブモデルから出力されることを特徴とする請求項３４に記載のシステムシミュレーション方法。

【請求項３６】前記周辺回路の検証は、前記割込み要求信号発生の要因となる事象の発生により起動される前記ハードウェアモデルシミュレーションによって行われることを特徴とする請求項３４または請求項３５に記載のシステムシミュレーション方法。

【請求項３７】前記割込み要求信号は前記シミュレーション装置の外部端子への通信データの受信により発生することを特徴とする請求項３６に記載のシステムシミュレーション方法。

【請求項３８】前記ハードウェアの検証が、前記シミュレーション装置の外部端子で受信した通信データのデータメモリへのＤＭＡ転送であることを特徴とする請求項１３～請求項１６のいずれかに記載のシステムシミュレーション方法。

【請求項３９】マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、
前記ターゲットプログラムはサブルーチン処理を有し、
前記サブルーチン処理を該サブルーチンによって得られる結果と等価な結果が得られる仮想モデルシミュレーションで置き換えて処理することを特徴とするシステムシミュレーション方法。

【請求項４０】マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、該システムシミュレーションの母体装置の読み出し専用メモリに、前記プログラムを書き込む手順と、前記母体装置のメモリに、前記プログラムを検証するマイコンシミュレーションプログラム，前記プログラムに基づいて前記ハードウェアを検証するハードウェアモデルシミュレーションプログラム，ＣＰＵ情報，ハードウェア接続情報，仮想モデル情報，仮想モデル選択情報およびシミュレータ間インタフェース情報を書き込んで該母体装置をシミュレーション装置化する手順と、前記プログラムにしたがってシステムシミュレーションを実行する手順とから成り、前記仮想モデル情報に前記ハードウェアの検証結果を含ませる手順と、前記ハードウェアの検証のうち前記仮想モデル選択情報で指定された部分については前記マイコンシミュレーションプログラムにより代替して実行する手順とを有することを特徴とするシステムシミュレーション方法。

【請求項４１】請求項１９～請求項４０のいずれかに記載のシステムシミュレーション方法を実行するプログラムを記録したコンピュータ読み込み可能なシステムシミュレーション記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、マイクロコンピュータを搭載した電子機器のシステムシミュレータおよびシステムシミュレーション方法に関する。

【０００２】

【従来の技術】電子機器のセットメーカーは、マイクロコンピュータ（以下、マイコンという）を搭載した電子

機器の開発において、新しい機能をセットに盛り込み他社との差別化を図ろうとすることが多い。このとき、セットメーカーは、開発期間を短縮するためには、新しいマイコンが完成する前に、電子機器に搭載するかどうかを検討する必要がある。

【０００３】しかしながら、セットメーカーは、新製品のマイコンのサンプルを入手しなければハードウェアの開発を開始できない。開発できたとしても、マイコンと周辺回路とのインタフェース部で不具合が発見され、電子機器の回路構成を再設計しなければならない場合も多い。

【０００４】また、マイコンと周辺回路を同一チップ上に組み込んだシステムＬＳＩにおいては、試作したシステムＬＳＩを評価してから、不具合を変更して再度試作したのでは開発期間が非常に延びてしまい。商機を逃しかねない。また、開発に要する費用も膨大になり、損失が大きい。

【０００５】そこで、ソフトウェアの検証を行うマイコンシミュレータと、ハードウェアを検証するハードウェアシミュレータを合体させて（このような合体されたシミュレータをシステムシミュレータという）、シミュレーション装置で動作させ、ハードウェアのプロトタイプが出来ていない開発初期の段階からソフトウェアとハードウェアの双方を同時に、しかも短期間に検証と修正をすることが極めて重要になってきた。

【０００６】マイコンシミュレータは、電子機器を構成するマイコンの機能をシミュレーションするＣＰＵモデルシミュレータと、電子機器を機能させるプログラム（以下、ターゲットプログラムという）を記憶するプログラムメモリとから構成される。このターゲットプログラムをＣＰＵモデルシミュレータ上で実行し、ターゲットプログラムが当初意図した順序で動作するか、あるいは、プログラムが暴走したり、止まってしまうことがないかなどを検証する。

【０００７】ハードウェアシミュレータは、ＣＰＵの周辺に接続したメモリやＩ／Ｏポート、論理回路（以下、周辺回路と記す）などが期待通りに動作するか否かを検証する。この検証を実行するため、ハードウェアを構成するロジックやトランジスタの接続情報や、これらの駆動能力、寄生容量、遅延時間など各種のパラメータがハードウェアシミュレータに記憶される。

【０００８】ハードウェアシミュレータは、シミュレーションするとき、ある周辺回路に所定の信号を入力し、その信号を接続情報をもとに周辺回路を構成するトランジスタに入力し、その出力を次のトランジスタに入力するといった処理を繰り返し、周辺回路からどのような信号がどのようなタイミングで出力されるかを演算によって求める。

【０００９】この演算量は膨大であるため、１つの信号を１つの周辺回路に入力し、その結果を得るのに数１０～１００ｍｓかかる。これに対し、ＣＰＵモデルシミュレータは、プログラムを１行実行するのに、数１０μｓで完了する。このように、ハードウェアシミュレータと組み合わせてソフトウェアのシミュレーションを実行する場合、ハードウェアシミュレータの演算速度でシミュレーション時間が決められてしまい、非常に長時間を要していた。

【００１０】例えば、通常のキースキャンは、出力ポートからキーマトリクスに信号を出力し、キーマトリクスからの信号を入力ポートを通して入力することで、押下されたキーを探す処理である。出力ポートにはキーマトリクスの行数または列数に相当する回数だけ信号を出力する必要がある。入力ポートについては、キーマトリクスの行数または列数を入力ポートの端子数で除算した回数だけ信号を入力する必要がある。マイコンシミュレータとハードウェアシミュレータ間で、１回の入力または出力処理で、後述のように、４回の通信が必要である。例えば、キーマトリクスが４行×４列の場合、８回の入出力を実行しなければならない。従って、１回のキースキャンあたり、通信回数は８×４＝３２回である。普通は、１回のキースキャンでキー押下を検出されることはなく、エンドレスで繰り返されるため、１０回繰り返したとしても、３２０回の通信が発生する。

【００１１】実際のキースキャン・プログラムは、チャタリング等による誤入力を防止するため、一度のキー押下を検出するだけではなく、一定時間経過後に再度キースキャンを実行し、同じキーが押され続けていることを確認し、押下されたキーを特定する。この処理をハードウェアシミュレータで行うと、前述のように数１００回～数１０００回の通信が必要であり、１つのキー押下を検出するだけで数分を要することになる。

【００１２】このため、従来はプログラムとハードウェアの検証を別々に実行したり、周辺回路は実際に組み立て、これをマイコンシミュレータに接続して検証するなどしていた。

【００１３】

【発明が解決しようとする課題】しかしながら、上述した従来のシミュレーション方法のうちの前者では、周辺回路に係わる命令のシミュレーションが十分に検証できず、電子機器に組み立ててから問題を発見することがあった。

【００１４】また、後者は、周辺回路を組み立てる工数や部品代が必要であり、問題が見つかったときは、再度組み立て直さなければならない。また、周辺回路をＩＣに組み込むとタイミングがずれることもあり、ＩＣ化したときに初めて問題に気付くこともあった。

【００１５】本発明の目的は、電子機器を構成するハードウェアの検証を、電子機器に組み込むプログラムの検証と同時に実行することで、検証時間の短縮と、検証内容の充実を図ったシステムシミュレータおよびシステム

シミュレーション方法を提供することにある。

【００１６】

【課題を解決するための手段】本発明のシステムシミュレータは、マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレータにおいて、前記プログラムに基づいて前記ハードウェアをソフトウェアで検証するハードウェアシミュレータと、前記ハードウェアに係わる前記プログラム命令を前記ハードウェアと等価的にソフトウェアで処理する仮想モデルシミュレータと、前記ハードウェアシミュレータまたは前記仮想モデルシミュレータの出力を適時に利用しながら前記プログラムをソフトウェアで検証するＣＰＵモデルシミュレータとを具備することを特徴とする。

【００１７】また、本発明のシステムシミュレーション方法は、マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、前記ハードウェアの接続情報に基づいて周辺回路の処理をシミュレーションする第１ステップと、前記ハードウェアの論理機能に基づいて周辺回路の処理を等価的にシミュレーションする第２ステップとを有し、前記第１または第２ステップのいずれかを選択してマイクロコンピュータの入出力命令をシミュレーションする第３ステップとを備えたことを特徴とする。

【００１８】

【発明の実施の形態】次に、本発明の実施の形態について説明する。

【００１９】本発明のシステムシミュレータは、マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレータにおいて、前記プログラムに基づいて前記ハードウェアをソフトウェアで検証するハードウェアシミュレータと、該ハードウェアの検証結果を適時に利用しながら前記プログラムをソフトウェアで検証するマイコンシミュレータとから成ることを特徴とする。

【００２０】また、本発明のシステムシミュレーション方法は、マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、該システムシミュレーションの母体装置の読み出し専用メモリに、前記プログラムを書き込む手順と、前記母体装置のメモリに、前記プログラムを検証するマイコンシミュレーションプログラム，前記プログラムに基づいて前記ハードウェアを検証するハードウェアシミュレーションプログラム，ＣＰＵ情報，ハードウェア接続情報，仮想モデル情報，仮想モデル選択情報およびシミュレータ間インタフェース情報を書き込んで該母体装置をシミュレーション装置化する手順と、前記プログラムにしたがってシステムシミュレーションを実行する手順とから成り、前記仮想モデル情報に前記ハードウェアの検証結果を含ませる手順と、前記ハードウェアの検証のうち前記仮想モデル選択情報で指定された部分については前記マイコンシミュレーションプログラムにより代替して実行する手順とを有することを特徴とする。

【００２１】以下、本発明の実施例について図面を参照して詳細に説明する。

【００２２】［第１実施例］

構成の概要

図１は本発明のシステムシミュレータの第１実施例を示す機能ブロック図である。本システムシミュレータは、マイコンシミュレータ１とハードウェアシミュレータ２とから成る。また、本シミュレータは、シミュレーションの母体装置となる、例えばワークステーションにロードされて実行されるが、図１におけるユーザインタフェース（ＧＵＩ）１８は、そのようなシミュレーション装置が提供するユーザインタフェースである。なお、これ以降、インタフェースはＩ／Ｆと記すこととする。

【００２３】ハードウェアシミュレータ２は、シミュレーション対象となる電子機器のハードウェアをソフトウェアで実現したハードウェアモデル２０を中核とし、本電子機器を制御するマイクロプロセッサ（以下ＣＰＵと記す）に接続される周辺回路の接続情報や各種パラメータなどが記憶されており、入力されたデータに対して、期待されるデータが、期待されるタイミングで出力されるか、すなわち、論理的な矛盾や動作タイミングの矛盾がないかを検証する。さらに、ハードウェアモデル２０は、図３にその具体例を示すように、各周辺回路の機能記述モデルから構成される。

【００２４】マイコンシミュレータ１は、検証するプログラムを１行単位に読み込んで、プログラムの記述や処理順序に問題ないかを検証し、ＣＰＵモデル４，機能関数Ｉ／Ｆモデル５，バスＩ／Ｆモデル６，仮想モデル７および機能関数選択テーブル３を有する。

【００２５】ＣＰＵモデル４は、ＣＰＵの機能をソフトウェアで実現したもので、次の処理を実行する。

・プログラムカウンタＰＣで指定されるメモリアドレスのデータを読み込む。

・データ（オペレーションコードあるいはオペランド）を解読する。

・解読結果に基づきその命令を実行する。

・周辺回路に関係する命令の場合、機能関数を生成する。

・機能関数Ｉ／Ｆモデル５に機能関数を渡す。

【００２６】機能関数Ｉ／Ｆモデル５は、ハードウェアシミュレータ２を使うか、仮想モデル７を使うか、その選択を判断し、選択された方に必要な情報を渡し、また選択された方から必要な情報を得る。得た結果はＣＰＵ

モデル４に渡す。なお、仮想モデル７を選択するかどうかの判断は、機能関数選択テーブル（具体例を図６に示す）３をもとに行う。機能関数選択テーブル３は、予めシステム開発者が作成し、本システムシミュレーション装置に組み込んであるものとする。

【００２７】仮想モデル７は、本発明の特徴部分の一つであり、本実施例では、ハードウェアモデル２０を構成する機能記述モデルのうち、予め設定されたものから成る。本実施例では、ある周辺回路のハードウェアシミュレーションの結果が他の周辺回路のシミュレーションに影響を及ぼさないものについてのみ仮想モデル７に登録できる。

【００２８】バスＩ／Ｆモデル６は、ハードウェアシミュレータ２のシステムＩ／Ｆモデル８との間で情報の受け渡しを行う。すなわち、機能関数Ｉ／Ｆモデル５がＣＰＵモデル４から受け取った機能関数をもとに、アドレス信号、データ信号、リード／ライト信号などをシステムＩ／Ｆモデル８を介してハードウェアモデル２０に渡し、ハードウェアシミュレータ２でシミュレーションした結果をシステムＩ／Ｆモデル８を介して得る。

【００２９】システムＩ／Ｆモデル８は、マイコンシミュレータ１のバスＩ／Ｆモデル６との間で情報の受け渡しを行う。バスＩ／Ｆモデル６を介して受け取ったアドレス信号、データ信号、リード／ライト信号などをもとに、ハードウェアシミュレータ２で演算を実行し、その結果をバスＩ／Ｆモデル６に返す。

【００３０】バスＩ／Ｆモデル６およびシステムＩ／Ｆモデル８は、またマイコンシミュレータ１とハードウェアシミュレータ２との間のコード変換をも行う。

【００３１】ユーザインタフェース１８は、電子機器のボタンや表示を画面に表示させて、画面上のボタンをマウス等で押した結果を読み取って、その情報をシミュレーション対象のプログラムに入力したり、プログラムで処理した結果を画面に表示させたりする。このＧＵＩ１８は、ハードウェアシミュレータ２だけでなく、仮想モデル７とも接続されており、双方と情報の授受を行う。

【００３２】図２は、本システムシミュレーションを行うための母体となる装置、例えばワークステーションをシミュレーション装置化するのに必要なプログラムファイルと情報ファイルを示す図である。本ワークステーションＥＷＳは、プロセッサＳ＿ＣＰＵ，読み出し専用メモリＳ＿ＲＯＭ，メモリＳ＿ＲＡＭ，割込制御装置Ｓ＿ＩＮＴ，ＤＭＡ装置Ｓ＿ＤＭＡ，表示装置と接続された表示制御装置Ｓ＿ＤＳＰ，キーボード，マウスと接続された入力部Ｓ＿ＩＮＰ，通信端末と接続されたシリアル入出力装置Ｓ＿ＳＩＯおよびハードディスクＳ＿ＨＤＤなどから構成される。

【００３３】ハードディスクＳ＿ＨＤＤには、マイコンシミュレータ１が動作するためのマイコンシミュレーションプログラム，ハードウェアシミュレータ２が動作するためのハードウェアシミュレータプログラム，マイコンの型番やレジスタ等の情報であるＣＰＵ情報，ハードウェアモデル２０についての前述のハードウェア接続情報，仮想モデル７に移植すべき周辺回路の機能記述モデルについての仮想モデル情報，マイコンシミュレーション時において仮想モデル７とハードウェアシミュレータ２のいずれを選択するかの基準となる仮想モデル選択情報およびマイコンシミュレータ１とハードウェアシミュレータ２とで情報の授受を行うときに必要なコード変換表等のインタフェース情報であるシミュレータ間インタフェース情報の各ファイルを格納する。

【００３４】これらの各ファイルはメモリＳ＿ＲＡＭにロードされ、このメモリＳ＿ＲＡＭ上に図１のマイコンシミュレータ１およびハードウェアシミュレータ２が仮想的に形成される。すなわち、ＣＰＵモデル４はマイコンシミュレーションプログラムによって動作し、ハードウェアシミュレータ２はハードウェアシミュレーションプログラムによって動作する。また、ＣＰＵモデル４とハードウェアモデル２０は、それぞれＣＰＵ情報とハードウェア情報によって構成される。さらに、仮想モデル７は仮想モデル情報にしたがって形成され、仮想モデル選択情報は機能関数選択テーブル３の内容となり、バスＩ／Ｆモデル６とシステムＩ／Ｆモデル８はシミュレータ間インタフェース情報によって構成される。

【００３５】なお、マイコンシミュレーションの対象となるプログラムは、マイコンシミュレーションプログラムの一部としてメモリＳ＿ＲＡＭにロードされる。また、図２の表示装置は、図１のＧＵＩ１８のＬＥＤ表示ＧＵＩ１８２やキー入力ＧＵＩ１８１を表示し、キーボードやマウスは、表示されたキー入力ＧＵＩ１８１のボタンを選択することにより、所望の情報をシミュレータに供給するためのものである。ここで、ＬＥＤ表示ＧＵＩ１８２は、表示装置の画面上に表示されたランプ画像であり、点灯したランプ画像と、消灯したランプ画像を有している。ＬＥＤ表示ＧＵＩ１８２は、供給される信号の状態に応じて、点灯または消灯の画像を表示する。また、キー入力ＧＵＩ１８１は、表示装置の画面上に表示されたスイッチ画像である。

【００３６】なお、上述のワークステーションＥＷＳの構成要素Ｓ＿ＨＤＤ、Ｓ＿ＤＭＡやファイルは、代表的な一例を示すものであって、用途に応じて適宜追加あるいは削除しうるものである。

【００３７】動作の概要

さて、以上のようにして構成された本シミュレータの動作の概要について説明する。ハードディスクＳ＿ＨＤＤには、図２の各ファイルが格納されており、また読み出し専用メモリＳ＿ＲＯＭにはシミュレーション対象のプログラム（ターゲットプログラム）が予め書き込まれているものとする。先ず、ワークステーションＥＷＳの電源を投入して起動後に、上述のマイコンシミュレーショ

ンプログラム，ハードウェアシミュレーションプログラム等をメモリＳ＿ＲＡＭにロードしてシミュレーション装置化する。すると、図１に示したようなマイコンシミュレータ１およびハードウェアシミュレータ２が構成され、システムシミュレーションの環境が整備されたので、いよいよシミュレーションを実行することになる。

【００３８】ＣＰＵモデル４は、シミュレーション対象プログラムをプログラムカウンタＰＣの値にしたがって順次に読み込む。ＣＰＵモデル４が読み出した命令がジャンプ命令ＢＲ，ＢＮＺ，減算命令ＳＵＢのように、周辺回路に関係がない場合、ＣＰＵモデル４内部でシミュレーションが実行される。一方、ＣＰＵモデル４が読み出した命令が周辺回路モデルに関係し、仮想モデル７に登録されていない場合、ＣＰＵモデル４は、生成した機能関数を、バスＩ／Ｆモデル６とシステムＩ／Ｆモデル８を介してハードウェアシミュレータ２に渡し、演算処理が実行され、その結果を得る。そして、ハードウェアシミュレータ２は、演算を実行し、その結果をマイコンシミュレータ１に返す。このような命令として、例えば、ある周辺回路のハードウェアシミュレーションの結果が他の周辺回路のシミュレーションに影響を及ぼすようなものを挙げることができる。

【００３９】ここで、機能関数とは、マイコンシミュレータ１とハードウェアシミュレータ２とではコード体系が異なることが少なくないことから、これらのコード体系にとらわれることなく取り扱えるように設定された中間的なコードをいう。

【００４０】一方、ＣＰＵモデル４が読み出した命令が周辺回路モデルに関係し、仮想モデル７に登録されている場合には、ＣＰＵモデル４は、生成された機能関数を、機能関数Ｉ／Ｆモデル５に渡し、仮想モデル７を参照し、所望の情報を得る。仮想モデル７は、プログラムを実行して、その結果を機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に返す。この場合には、マイコンシミュレータ１は、周辺回路モデルに関わる命令を実行するのに、ハードウェアシミュレータ２ではなく、仮想モデル７から命令の実行結果を受け取ることになる。

【００４１】これにより、マイコンシミュレータ１とハードウェアシミュレータ２との間での情報のやりとりが低減でき、シミュレータ相互間の情報の受け渡し回数や待ち合わせ時間を削減できるのである。

【００４２】〔構成の具体例〕次に、本実施例の具体例としてマイコンを使用した電子手帳のシステムシミュレーションをとり挙げ、より詳細に説明する。

【００４３】図３は、この電子手帳の一部分を構成するハードウェアモデル２０の詳細を示すブロック図であり、プログラムを記憶するプログラムメモリモデル１０，データを記憶するデータメモリモデル１１，電子手帳の時計を模擬する時計モデル１７，電子手帳の押ボタン群への入力機能モデルであるキー入力モデル１９およびＧＵＩ１８への出力ポートを模擬するＩ／ＯポートＣモデル１６とで構成される。

【００４４】キー入力モデル１９は、電子手帳の押ボタン群を模擬した４行４列のキーマトリクスモデル１３，デコーダモデル１５への入力信号ＰＢ0 ，ＰＢ1 を与えるＩ／ＯポートＢモデル１４，この入力信号を解読して、キーマトリクスモデル１３の４本の列線のうちのいずれか１本の列線を活性化する信号Ｄ0 ～Ｄ3 を出力するデコーダモデル１５，キーマトリクスモデル１３で押下された接点を含む行線上の信号ＰＡ0 ～ＰＡ3 を入力するＩ／ＯポートＡモデル１２から成る。

【００４５】プログラムメモリモデル１０，データメモリモデル１１，Ｉ／ＯポートＡモデル１２，Ｉ／ＯポートＢモデル１４，デコーダモデル１５，時計モデル１７の各ハードウェアモデル（周辺回路対応のサブモデル）は、図２に示したハードウェアシミュレーションプログラムおよびハードウェア接続情報に基づいて、演算するための回路接続情報と各種パラメータをメモリＳ＿ＲＡＭに記憶している。また、プログラムメモリモデル１０は、ターゲットプログラムを記憶するテーブル（記憶領域）を有している。

【００４６】また、プログラムメモリモデル１０，データメモリモデル１１，時計モデル１７，Ｉ／ＯポートＡモデル１２，Ｉ／ＯポートＢモデル１４およびＩ／ＯポートＣモデル１６は、バスＢＵＳを介してシステムＩ／Ｆモデル８と接続される。時計モデル１７は、ＣＰＵモデル４に供給されるクロック信号ＨＣＬＫ（不図示）が供給され、これを計数した時刻情報を保持している。

【００４７】デコーダモデル１５からの信号Ｄ0 ～Ｄ3 およびＩ／ＯポートＣモデル１６からの信号ＰＣ0 は、ＧＵＩ１８へ導かれ、ＧＵＩ１８からの信号ＰＡ0 ～ＰＡ3がＩ／ＯポートＡモデル１２へのキーマトリクスモデル１３側からの入力となる。すなわち、キーマトリクスモデル１３はあくまで仮想的なものであるため、人の現実の操作はＧＵＩ１８上で行われる。したがって、デコーダモデル１５からの信号Ｄ0 ～Ｄ3 は実際にはＧＵＩ１８に入力し、表示装置にマトリクス状に配置された複数のキースイッチ（キー入力ＧＵＩ１８１）が表示され、人がキーボードまたはマウスでＧＵＩ１８上のキー入力ＧＵＩ１８１のいずれかを選択することで、キーマトリクスモデル１３からの信号ＰＡ0 ～ＰＡ3 となってＩ／ＯポートＡモデル１２に入力するものである。

【００４８】図４は、図３のハードウェアモデル２０に対応する仮想モデル７の詳細な構成例を示し、プログラムメモリモデル１０対応の仮想プログラムメモリモデルＶ７１，時計モデル１７対応の仮想時計モデルＶ７２，Ｉ／ＯポートＡモデルとＩ／ＯポートＢモデル対応の仮想キー入力モデルＶ７３、およびＩ／ＯポートＣモデル対応の仮想ＬＥＤ表示モデルＶ７４から成る。このような仮想モデル７の構成は、図２における仮想モデル情報

ファイルをワークステーションＥＷＳにロードすることにより、実現されるものである。

【００４９】本実施例では、前述のように、ある周辺回路のハードウェアシミュレーションの結果が、他の周辺回路のシミュレーションに影響を及ぼさないものについてのみ仮想モデル７に登録できる。このことから、時計モデル１７とＩ／ＯポートＣモデル１６は、他の周辺回路とは独立して動作しており、ＣＰＵモデル４以外とデータの授受がないので仮想モデル７に登録できるのに対し、Ｉ／ＯポートＢモデル１４のシミュレーション結果は、Ｉ／ＯポートＡモデル１２に影響を及ぼすので、どちらか一方だけを仮想モデル７に登録することはできないのである。

【００５０】仮想プログラムメモリモデルＶ７１は、図５に例示するように、プログラムメモリモデル１０に記憶されたターゲットプログラムの命令コードと同じ命令コードをテーブル（記憶手段）に予め記憶している。このテーブルには、仮想プログラムメモリモデルＶ７１のアドレスに対応するシンボルと命令コードに相当する戻り値とが配置されている。

【００５１】仮想時計モデルＶ７２は、ＣＰＵモデル４に供給されるクロックＶＣＬＫ（不図示）を計数し、時刻情報を保持している。クロックの計数処理は、予め登録されたプログラムに従って処理され、マイコンシミュレータ１が他の処理を実行中であっても常に実行されており、マイコンを実使用したときに相当する時刻を出力する。ハードウェアシミュレータ２は、シミュレーションに時間がかかるため、クロックを計数しきれないが、マイコンシミュレータ１は、ハードウェアシミュレータ２により処理速度が速いので、クロックを計数することができる。計数結果は現実の時刻とは違い、進み方が遅いが、マイコンを実際に動作させたときには現実の時刻になるように設定されているので、シミュレーション実行中に実際に使用したときに相当する時刻を知ることができる。

【００５２】仮想キー入力モデルＶ７３は、キー入力ＧＵＩ１８１のサブルーチンが呼び出されたとき、仮想モデル７内に予め登録したキー入力サブルーチンを処理することで、ＧＵＩ１８上で押されたボタンに相当する文字情報、例えば「０」、「１」、「Ａ」などを得ることができる。これに対して、ハードウェアモデル２０のデコーダモデル１５からは、バイナリ形式の［０，１，０，０］をＧＵＩ１８のキー入力ＧＵＩ１８１に供給し、キー入力ＧＵＩ１８１からは、バイナリ形式の入力信号ＰＡ0 ～ＰＡ3 ［０，０，１，０］をＩ／ＯポートＡ１２に受給することで、ＣＰＵモデル４はターゲットプログラム内のキー入力サブルーチンを処理して、どのボタンが押されたかを判断する。

【００５３】仮想ＬＥＤ表示モデルＶ７４は、供給されたＬＭＰ＊＊をもとに、ＧＵＩ１８のＬＥＤ表示ＧＵＩ１８２に供給する信号に変換する処理を実行する。例えば、「ＬＭＰ００」が呼び出されたとき、仮想ＬＥＤ表示モデルＶ７４は「０」を出力し、「ＬＭＰ０１」のときには「１」を出力する。

【００５４】ＧＵＩ１８のＬＥＤ表示ＧＵＩ１８２は、ＬＥＤが点灯した画像情報と、ＬＥＤが消灯した画像情報とを有しており、仮想ＬＥＤ表示モデルＶ７４の指示に応じて、いずれか一方の画像情報を選択して表示装置に表示する。「０」が入力されたときには、ＬＥＤが消灯した画像がＬＥＤ表示ＧＵＩ１８２に表示され、「１」が入力されたときには、ＬＥＤが点灯した画像がＬＥＤ表示ＧＵＩ１８２に表示される。

【００５５】これに対して、ＣＰＵモデル４からハードウェアモデル２０のＩ／ＯポートＣモデル１６にＩ／Ｏポート出力命令が実行され、ポートＰＣ0 が「０」になれば、ＬＥＤが消灯した画像がＬＥＤ表示ＧＵＩ１８２に表示され、「１」になれば、ＬＥＤが点灯した画像がＬＥＤ表示ＧＵＩ１８２に表示さる。

【００５６】図６は、図１に示した機能関数選択テーブル３の内容例であり、機能関数Ｉ／Ｆモデル５が、仮想モデル７を選択可能な機能関数とその判断条件を示す。ＣＰＵモデル４から渡された機能関数が次の場合に、機能関数Ｉ／Ｆモデル５は仮想モデル７を選択する。ＣＰＵモデル４から渡された機能関数が機能関数選択テーブル３に記述されていなかったり、あるいは、記述されていても、判断条件を満たさない場合には、機能関数Ｉ／Ｆモデル５はハードウェアモデル２０を選択して検証する。

【００５７】図６において、ＡＤＲＳはＣＰＵモデル４がアクセスするアドレスを示す信号であり、Ｉ／Ｏはメモリを除く周辺回路の入出力命令であることを示す信号であり、ＲＤは読み出しであることを示す信号であり、ＷＲは書き込みであることを示す信号である。また、「ＲＥＡＤＴＩＭＥ」、「ＰＯＲＴＣ」、「ＫＥＹＩＮ」は、アドレスを示すラベルである。また、「Ｄ＊」は任意のデータを意味し、「ＴＩＭＥ」は時計モデル１７または仮想時計モデルＶ７２から読み出した時刻を意味する。また、「＆」は、条件の論理積、即ち、「＆」の前後に記述された条件が同時に満たされるとき、判定条件が有効であることを意味する。

【００５８】仮想モデル７を選択可能な機能関数とその判断条件は次の通りである。

【００５９】■ＦＥＴＣＨ関数：仮想プログラムメモリモデルＶ７１から命令を読み込む関数であり、引数は、アドレスＡＤＲＳである。本例では、アドレスＡＤＲＳが１０００Ｈ以上で、２０００Ｈ未満なら仮想モデル７を選択可能。

【００６０】■ＩＯＲＤ関数：メモリ以外の周辺回路を読み出す関数で、引数はアドレスＡＤＲＳと、Ｉ／Ｏと、ＲＤである。本例では、アドレスＡＤＲＳが「ＲＥ

ＡＤＴＩＭＥ」であるときは、仮想モデル７を選択可能。

【００６１】■ＩＯＷＲ関数：メモリ以外の周辺回路にデータを書き込む関数で、引数は、アドレスＡＤＲＳ、書き込みデータ「Ｄ＊」、Ｉ／Ｏ、及びＷＲである。本例では、アドレスＡＤＲＳが「ＰＯＲＴＣ」で、データは任意の値「Ｄ＊」であれば選択可能。

【００６２】■ＣＡＬＬ関数：特定のサブルーチンを呼び出したときに生成される関数で、引数はサブルーチンの先頭アドレスＡＤＲＳと、時刻ＴＩＭＥである。本例では、アドレスＡＤＲＳがラベルＫＥＹＩＮで、時刻が０５Ｈ以上で１５Ｈ以下の場合は、仮想モデル７を選択可能。

【００６３】図６に示す機能関数選択テーブル３は一つの例であって、このテーブルに登録する機能関数は、シミュレーションする対象に応じて適宜書き換えることができる。また、引数の条件、およびその組み合わせも適宜書き換えることができる。例えば、ＣＡＬＬ関数において、時刻ＴＩＭＥの条件は必須ではない。また、Ｉ／Ｏ、ＲＤ、ＷＲなども、機能関数名ＩＯＲＤ、ＩＯＷＲにより判断できる場合には無くてもよい。

【００６４】図７は、機能関数Ｉ／Ｆモデル５における仮想関数変換テーブルと、その戻り値を示す。機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４からは、前述のような理由から機能関数を受け取る。しかし、ＣＰＵモデル４と仮想モデル７とは異なるソフトメーカーで開発されることがあり、ＣＰＵモデル４が使用している機能関数コードと、仮想モデル７が使用する仮想関数コードとは異なることが多い。そこで、ＣＰＵモデル４対応のコード（機能関数コード）と仮想モデル７対応のコード（仮想関数コード）とを本テーブルにより変換することにしたのである。なお、この仮想関数変換テーブルは、図２のＣＰＵ情報および仮想モデル情報により生成される。また、図７では、引数［メモリ、Ｉ／Ｏ］＆［ＲＤ、ＷＲ］は省略している。

【００６５】図８は、プログラムメモリモデル１０および仮想プログラムメモリモデルＶ７１のアクセス・アドレスと、そのアドレスに記憶された命令コード、および、その命令コードの意味を示すニーモニック表示とを表にしたものである。

【００６６】図８のターゲットプログラムリストは、電子手帳のボタン押下の正当性を検証するものであり、キー入力ＧＵＩ１８１に対して、あるボタンが押されればＬＥＤ表示ＧＵＩ１８２を点灯表示し、他のボタンが押されれば、ＬＥＤ表示ＧＵＩ１８２を消灯表示するというものである。

【００６７】なお、プログラムリスト中、＃はイミーディエト・データ指定、！は絶対アドレス指定、Ｈは１６進数であることを示す。

【００６８】図８に示すターゲット・プログラム・リストの１行目「１００１」番地の「Ａ２０１」は、キーセンスを行うために、Ｉ／ＯポートＢに「０１Ｈ」出力する命令、２行目「１００２」番地の「Ｃ５０１」は、Ｉ／ＯポートＡからデータを入力してレジスタＡに格納する命令、３行目「１００３」番地の「Ｂ２０４」は、レジスタＡから「０１Ｈ」を減算する命令、４行目「１００４」番地の「７４０６」は、レジスタＡが「０Ｈ」であれば「１００６Ｈ」番地に分岐する命令、５行目「１００５」番地の「２００２」は、無条件で「１００２Ｈ」番地に分岐する命令、６行目「１００６」番地の「５ＡＢ５」は、時計モデルからデータを読み込んでレジスタＡに格納する命令、７行目「１００７」番地の「Ｂ３１０」は、レジスタＡから「１０Ｈ」を減算する命令、８行目「１００８」番地の「７４０Ａ」は、レジスタＡが「０Ｈ」であれば「１００ＡＨ」番地に分岐する命令、９行目「２００６」は、無条件で「１００６Ｈ」番地に分岐する命令、１０行目「１００Ａ」番地の「Ｃ７０１」は、Ｉ／ＯポートＣに「０１Ｈ」を出力する命令、１１行目「１００Ｂ」番地の「５Ａ０３」は、サブルーチン「ＫＥＹＩＮ」を呼び出す命令、１２行目「１００Ｃ」番地の「Ｂ２ＥＣ」は、レジスタＡから「ＥＣＨ」を減算する命令、１３行目「１００Ｄ」番地の「７４０Ｆ」は、レジスタＡが「０Ｈ」であれば「１００ＦＨ」番地に分岐する命令、１４行目「１００Ｅ」番地の「２００Ｂ」は、無条件で「１００ＢＨ」番地に分岐する命令、１５行目「１００Ｆ」番地の「Ｃ７００」は、Ｉ／ＯポートＣに「００Ｈ」を出力する命令である。

【００６９】また、１７行目の「ＫＥＹＩＮ」はラベルを表し、２４行目「３０１Ｆ」番地の「ＲＥＴ」はサブルーチンが呼び出された番地に戻ることを表す。また、１７行目から２４行目までのリストは、キー入力のサブルーチンが記述されており、キーマトリクスモデル１３をキースキャンして、押されたキーに対応付けられた文字コードを戻すものとする。

【００７０】図９は、ＣＰＵモデル４のシミュレーションプログラムのフローチャートを示す。

【００７１】図９において、ステップＳ１０１で、ＣＰＵモデル４は、プログラムカウンタＰＣの値を取得する。ＣＰＵモデル４は、プログラムカウンタＰＣの値を取得後、次の取得に備えてプログラムカウンタＰＣの値を＋１だけインクリメントする。

【００７２】ステップＳ１０２で、ＣＰＵモデル４は、プログラムカウンタＰＣで示されるアドレスをアクセスして、プログラムメモリモデルから命令コードを読み出すため、機能関数ＦＥＴＣＨ（ＰＣ）を生成する。ここで、ＰＣはプログラムカウンタＰＣの値である。

【００７３】ステップＳ１０３で、ＣＰＵモデル４は、メモリを含む周辺回路をアクセスして所定の処理を実行する周辺回路処理サブルーチンを呼び出す。ここでは、

機能関数ＦＥＴＣＨ（ＰＣ）を機能関数Ｉ／Ｆモデル５に渡して、プログラムメモリモデルから命令コードを読み出す処理を実行する。

【００７４】ステップＳ１０４で、ＣＰＵモデル４は、周辺回路処理サブルーチンにより得られた命令コードを命令レジスタに格納する。

【００７５】ステップＳ１０５で、ＣＰＵモデル４は、命令レジスタに格納された命令コードを解読する。

【００７６】ステップＳ１０５で、ＣＰＵモデル４は、解読された命令コードが、ロード命令またはストア命令であって、周辺回路との入出力（読み書き）が生ずるか否かを判断する。周辺回路との入出力が生ずる場合には「Ｙ」へ進み、ステップＳ１０７の処理を実行する。周辺回路との入出力が生じない場合には「Ｎ」へ進み、ステップＳ１０９の処理を実行する。

【００７７】ステップＳ１０７で、ＣＰＵモデル４は、解読された命令コードに基づき周辺回路に対してデータを入力または出力する処理を実行するため、ロード命令またはストア命令に対応する機能関数を生成する。

【００７８】ステップＳ１０８で、ＣＰＵモデル４は、メモリを含む周辺回路をアクセスして所定の処理を実行する周辺回路処理サブルーチンを呼び出す。このとき、ＣＰＵモデル４は、機能関数Ｉ／Ｆモデル５に機能関数を渡すことにより、周辺回路に対するロード命令またはストア命令が処理される。このサブルーチンは、ステップＳ１０３で呼び出されるサブルーチンと別であってもよいが、ここでは同一のサブルーチンを呼び出すものとして説明する。

【００７９】ステップＳ１０９で、ＣＰＵモデル４は、ＣＰＵモデル４内で処理される分岐命令や加減算命令などを実行したり、周辺回路処理サブルーチンにより得られたデータを所定のレジスタに格納したりするなどの処理を実行する。

【００８０】ステップＳ１１０で、ＣＰＵモデル４は、実行した命令が「ＢＲＥＡＫ」や「ＳＴＯＰ」であれば「Ｙ」へ進み、シミュレーションの処理を一旦停止したり終了したりする。そうでなければ「Ｎ」へ進み、ステップＳ１０１へ戻り、ステップＳ１０１からＳ１０９の処理を繰り返す。

【００８１】図１０は、図９における周辺回路処理サブルーチンＳ１０３、Ｓ１０８の詳細フローチャートを示す。

【００８２】図１０において、ステップＳ２０１で、機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４から機能関数を受け取り、この機能関数が機能関数選択テーブル３の条件を満たすか否かを判断する。即ち、仮想モデル７を使って検証するか、ハードウェアシミュレータ２を使ってシミュレーションするかを判断する。機能関数選択テーブル３の判断条件を満たし、仮想モデル７を使う場合には、「Ｙ」へ進み、ステップＳ２０２を実行する。機能関数選択テーブル３の判断条件を満たし、仮想モデル７を使う場合には、「Ｙ」へ進み、ステップＳ２０２を実行する。そうでなければ、「Ｎ」へ進み、ステップＳ２０３を処理する。このとき、機能関数Ｉ／Ｆモデル５は、仮想モデル７、または、バスＩ／Ｆモデル６に機能関数を渡す。

【００８３】ステップＳ２０２で、仮想モデル７は、機能関数Ｉ／Ｆモデル５より機能関数を受け取り、後述の仮想モデル・シミュレーションを実行する。実行後、サブルーチンが呼び出されたステップＳ１０３または、Ｓ１０８の次のステップへ戻る。

【００８４】ステップＳ２０３で、バスＩ／Ｆモデル６は、機能関数Ｉ／Ｆモデル５より受け取った機能関数を、ハードウェアシミュレータ２のシミュレーションに必要な端子信号コード（ＣＳ、ＡＤＲＳ、ＤＡＴＡなど）に変換し、所定のマシンサイクルでシステムＩ／Ｆモデル８に渡す。

【００８５】ステップＳ２１０で、バスＩ／Ｆモデル６は、端子信号コードをシステムＩ／Ｆモデル８に渡すとき、端子信号コード名称の変換を行ったり、信号授受のタイミングを調整したりする。一般に、シミュレーションを実現するソフトウェアは、複数のベンダから供給されることが多く、シミュレータ間のコード体系の違いを吸収したり、信号授受のタイミングの違いを調整する必要がある。このような問題がない場合には、ユーザはステップ２１０を省略してもよい。

【００８６】ステップ２１１で、システムＩ／Ｆモデル８は、バスＩ／Ｆモデル６から端子信号コードを受け取り、これをハードウェアシミュレータ２に渡す。

【００８７】ステップ２０９で、ハードウェアシミュレータ２は、受け取った端子信号コードをもとにシミュレーションを実行し、その結果をシステムＩ／Ｆモデル８に渡す。

【００８８】ステップ２１１で、システムＩ／Ｆモデル８は、ハードウェアシミュレータ２からシミュレーション結果を受け取り、バスＩ／Ｆモデル６にこれを渡す。

【００８９】ステップＳ２１０で、バスＩ／Ｆモデル６は、システムＩ／Ｆモデル８からシミュレーション結果コードを受け取るとき、結果コード名称の変換を行ったり、信号授受のタイミングを調整したりする。

【００９０】ステップＳ２０３で、バスＩ／Ｆモデル６は、システムＩ／Ｆモデル８から受け取った結果コードを、機能関数Ｉ／Ｆモデル５に渡し、機能関数Ｉ／Ｆモデル５はこれをＣＰＵモデル４に渡す。

【００９１】以上で、一連の周辺回路処理サブルーチンを終了し、サブルーチンが呼び出されたステップＳ１０３または、Ｓ１０８の次のステップへ戻る。

【００９２】次に、バスＩ／Ｆモデルシミュレーション Ｓ２０３とシステムＩ／Ｆモデルシミュレーション内の詳細な動作を説明する。

【００９３】ステップＳ２０４で、バスＩ／Ｆモデル６は、機能関数Ｉ／Ｆモデル５から受け取った機能関数を、ハードウェアシミュレータ２のシミュレーションに必要な端子信号コード（ＣＳ、ＡＤＲＳ、ＤＡＴＡなど）に変換し、所定のマシンサイクルでシステムＩ／Ｆモデル８に渡す準備を行う。

【００９４】図１２に、バスＩ／Ｆモデル６がハードウェアシミュレータ２と授受する代表的な端子信号コードと、その信号変化を表すタイミングチャートを示す。

【００９５】図１２において、マシンサイクルは、マイコンのクロックに相当し、ここでは、４マシンサイクルで１つのロード命令またはストア命令に基づく端子信号コードが授受されるものとする。／ＣＳはチップ選択端子信号，ＡＤＲＳはアドレス端子信号，／ＲＤはリード端子信号，／ＷＲはライト端子信号，ＤＡＴＡはデータ端子信号である。なお、「／」は信号がローレベル“０”のとき有効であることを意味する。

【００９６】マシンサイクルＭ１の時点では、チップ選択端子信号／ＣＳと、アドレス端子信号ＡＤＲＳとが出力され、マシンサイクルＭ２では、リード端子信号／ＲＤまたはライト端子信号／ＷＲと、データ端子信号ＤＡＴＡとが出力される。マシンサイクルＭ３では、リード端子信号／ＲＤまたはライト端子信号／ＷＲと、データ端子信号ＤＡＴＡとが非活性化された信号が出力される。マシンサイクルＭ４では、チップ選択端子信号／ＣＳと、アドレス端子信号ＡＤＲＳとが活性化された信号が出力され、端子信号の授受が終了する。

【００９７】なお、本実施例では、システムＩ／Ｆモデル８は、第１～第４マシンサイクルの端子信号コードの変化を一括して受け取り、ハードウェアモデルシミュレーションＳ２０９を行った結果をバスＩ／Ｆモデル６に返すものとして説明する。従って、読み出しデータＤＡＴＡは第２マシンサイクルでは出力されず、シミュレーション結果を返すタイミングで出力される。別の方法として、ハードウェアシミュレータ２は、１マシンサイクル分の端子信号コードを受け取る毎にハードウェアモデルシミュレーションＳ２０９を実行することもできる。この方法であれば、システムＩ／Ｆモデル８は、第２マシンサイクルで読み出しデータＤＡＴＡを出力することができる。

【００９８】ステップＳ２０５で、バスＩ／Ｆモデル６は、ステップＳ２１０を介してシステムＩ／Ｆモデル８に第１マシンサイクルから順番に第４マシンサイクルまで端子信号コードを出力する。

【００９９】ステップＳ２１３で、システムＩ／Ｆモデル８は、複数の端子信号コードの変化を判断し、変化がなければ、「Ｎ」に進み、ステップＳ２１３に戻り変化を待つ。複数の端子信号コードのうちいずれか１つが変化すれば、入力データの準備ができたと判断し、「Ｙ」へ進み、ステップＳ２１４に移る。

【０１００】ステップＳ２１４で、システムＩ／Ｆモデル８は、バスＩ／Ｆモデル６から１マシンサイクル分の端子信号コードを受け取る。

【０１０１】ステップＳ２１５で、システムＩ／Ｆモデル８は、バスＩ／Ｆモデル６から４マシンサイクル分の端子信号コードの受領が完了したかを判断する。未完了であれば、「Ｎ」へ進み、ステップＳ２１２に移り、完了すれば、「Ｙ」へ進み、ステップＳ２０９の処理に移る。

【０１０２】ステップＳ２０９で、ハードウェアモデル２.０は、受領した端子信号コードをもとにハードウェアモデルシミュレーションを実行し、結果をシステムＩ／Ｆモデル８に返す。

【０１０３】ステップＳ２１２で、システムＩ／Ｆモデル８は、ステップＳ２１０を介してバスＩ／Ｆモデル６にシミュレーション結果または端子信号コード受領結果を返す。システムＩ／Ｆモデル８は、端子信号コードを正確に受領すると受信確認信号ＡＣＫを返し、正確に受領できないときは再送信を要求するため受信不能信号ＮＡＣＫを返し、ロード命令の実行後でシミュレーション結果があるときは出力の端子信号コードを返し、ストア命令の実行後でシミュレーション結果がないときはシミュレーション完了信号ＤＯＮＥを返す。

【０１０４】ステップＳ２０６で、バスＩ／Ｆモデル６は、システムＩ／Ｆモデル８から前記端子信号の変化があったか否かを判断する。変化がなければ、「Ｎ」に進み、ステップＳ２０６に戻り変化を待つ。複数の端子信号コードのうちいずれか１つが変化すれば、入力データの準備ができたと判断し、「Ｙ」へ進み、ステップＳ２０７に移る。

【０１０５】ステップＳ２０７で、バスＩ／Ｆモデル６は、システムＩ／Ｆモデル８から端子信号コードまたはシミュレーション結果を受け取る。

【０１０６】ステップＳ２０８で、バスＩ／Ｆモデル６は、システムＩ／Ｆモデル８から受領したコードをもとに、４マシンサイクル分の端子信号コードの送信が完了したか、あるいは、シミュレーションが完了したか否かを判断する。未完了あるいはＮＡＣＫであれば、「Ｎ」へ進み、ステップＳ２０５に移る。完了すれば、「Ｙ」へ進み、一連の周辺回路処理サブルーチンを終了し、サブルーチンが呼び出されたステップＳ１０３または、Ｓ１０８の次のステップへ戻る。

【０１０７】図１１は、図１０における仮想モデルシミュレーションＳ２０２の詳細フローチャートを示す。

【０１０８】図１１において、ステップＳ３０１で、仮想モデル７は、機能関数Ｉ／Ｆモデル５から機能関数を受け取ると、機能関数を仮想モデル７のコード体系にあう仮想関数に変換する（図７）。例えば、ＦＥＴＣＨ（ｍ）はＭＥＭｍに、ＩＯＲＤ（ＲＥＡＤＴＩＭＥ）はＲ＿ＴＭに、ＩＯＷＲ（ＰＯＲＴＣ、ｎ）はＬＭＰｎ

に、ＣＡＬＬ（ＫＥＴＩＮ）はＫ＿＿ＩＮにそれぞれ変換される。なお、コード体系が同じであれば、本ステップは省略してもよい。

【０１０９】ステップＳ３０２で、仮想モデル７は、仮想関数がＦＥＴＣＨ関数ＭＥＭｍであるかを判断し、ＭＥＭｍであれば、「Ｙ」へ進み、ステップＳ３０７へ移り、ＭＥＭｍでなければ、「Ｎ」へ進み、ステップＳ３０３へ移る。

【０１１０】ステップＳ３０３で、仮想モデル７は、仮想関数が時刻読出関数Ｒ＿＿ＴＭであるかを判断し、Ｒ＿＿ＴＭであれば、「Ｙ」へ進み、ステップＳ３０８へ移り、Ｒ＿＿ＴＭでなければ、「Ｎ」へ進み、ステップＳ３０４へ移る。

【０１１１】ステップＳ３０４で、仮想モデル７は、仮想関数がキー入力関数Ｋ＿＿ＩＮであるかを判断し、Ｋ＿＿ＩＮであれば、「Ｙ」へ進み、ステップＳ３０９へ移り、Ｋ＿＿ＩＮでなければ、「Ｎ」へ進み、ステップＳ３０５へ移る。

【０１１２】ステップＳ３０５で、仮想モデル７は、仮想関数がＬＥＤ点灯／消灯関数ＬＭＰであるかを判断し、ＬＭＰであれば、「Ｙ」へ進み、ステップＳ３１０へ移り、ＬＭＰでなければ、「Ｎ」へ進み、ステップＳ３０６へ移る。

【０１１３】ステップＳ３０６で、仮想モデル７は、受領した仮想関数が登録されていない関数であったり、パラメータが違っていたりすると、表示装置にエラーメッセージを表示する。

【０１１４】ステップＳ３０７で、仮想モデル７内の仮想プログラムメモリモデルＶ７１（図４）は、予め登録されたターゲットプログラムのテーブル（図５）を参照し、シンボルＭＥＭｍに対応する戻り値を読み出し、機能関数Ｉ／Ｆモデル５に返し、処理を終了する。

【０１１５】ステップＳ３０８で、仮想モデル７内の仮想時計モデルＶ７２（図４）は、読み出し要求のあった時点の時刻を戻り値として機能関数Ｉ／Ｆモデル５に返し、処理を終了する。なお、仮想時計モデルＶ７２は、予め登録された時計処理プログラムを有しており、このプログラムに従ってクロック信号ＶＣＬＫを常に計数し、時刻情報を生成する。

【０１１６】ステップＳ３０９で、仮想モデル７内の仮想キー入力モデルＶ７３（図４）は、キー入力ＧＵＩ１８１に対してキー入力を要求し、押下されたキーの文字情報を取得し、その文字情報を戻り値として機能関数Ｉ／Ｆモデル５に返し、処理を終了する。なお、本実施例では、キー入力ＧＵＩ１８１により文字情報を取得する例を示すが、図２に示すキーボードで押されたキーを検出し、入力部Ｓ＿＿ＩＮＰを介して押下されたキーの文字情報を取得するようにしてもよい。

【０１１７】ステップＳ３１０で、仮想モデル７内の仮想ＬＥＤ表示モデルＶ７４（図４）は、受領した機能関数Ｉ／Ｆモデル５をもとに、ＬＥＤ表示ＧＵＩ１８２にランプの点灯画像、または消灯画像を表示させる。例えば、仮想関数がＬＭＰ００であれば、ＬＥＤ表示ＧＵＩ１８２に「０」を出力して消灯表示にし、ＬＭＰ０１であれば、ＬＥＤ表示ＧＵＩ１８２に「１」を出力して点灯表示にする。処理が完了すると、機能関数Ｉ／Ｆモデル５に完了信号ＤＯＮＥを返し、処理を終了する。

【０１１８】図１１に示す上述の処理が終了すると、図１０のステップＳ２０２の次のステップ「戻る」に戻る。

【０１１９】［動作の詳細］次に、図８に示したターゲットプログラムのリストに基づいて、本電子手帳のシステムシミュレーションの動作につき、図５のプログラムメモリテーブル内のテーブル，図９～図１１のフローチャートおよび図１２のタイミングチャートを参照しながら、１つのプログラムステップごとに詳述する。

【０１２０】《ＲＯＭアドレス１００１Ｈの処理》図９のステップＳ１０１で、ＣＰＵモデル４は、プログラムカウンタＰＣの値「１００１Ｈ」を取得する。

【０１２１】ステップＳ１０２で、ＣＰＵモデル４は、機能関数ＦＥＴＣＨ（１００１Ｈ）を生成し、機能関数モデル５に渡す。

【０１２２】ステップＳ１０３で、これにより、周辺回路処理サブルーチンが呼出される。

【０１２３】図１０のステップＳ２０１で、機能関数Ｉ／Ｆモデル５は、機能関数選択テーブル３（図６）に関数が登録されていて、且つその引数の条件を満たしているか、チェックを行う。今の場合、ＦＥＴＣＨ関数は登録されており、アドレスの条件を満たしているので、機能関数を仮想モデル７渡し、「Ｙ」に進む。

【０１２４】ステップＳ２０２で、仮想モデル７は、仮想モデルシミュレーションを実行する。

【０１２５】図１１のステップＳ３０１で、仮想モデル７は、機能関数Ｉ／Ｆモデル５より受領した機能関数ＦＥＴＣＨ（１００１Ｈ）を、仮想関数変換テーブル（図７）をもとに仮想関数ＭＥＭ１００１に変換しする。

【０１２６】ステップＳ３０２で、仮想モデル７は、変換した仮想関数がプログラムメモリを読み出すための関数（ＭＥＭ関数）かを判断する。今の場合、ＭＥＭ関数なので、「Ｙ」へ進む。

【０１２７】ステップＳ３０７で、仮想モデル７は、仮想プログラムメモリモデルＶ７１（図４）内のテーブル（図５）を参照し、ラベル「ＭＥＭ１００１」に対応する戻り値「Ａ２０１」を取得し、機能関数Ｉ／Ｆモデル５にこれを返す。

【０１２８】図１１と図１０の処理を終了し、図９に戻る。

【０１２９】ステップＳ１０４で、機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４に戻り値「Ａ２０１」を渡すとともに、ＣＰＵモデル４は、戻り値「Ａ２０１」を読み

込む。

【０１３０】ステップＳ１０５で、ＣＰＵモデル４は、戻り値「Ａ２０１」を解析し、「ＭＯＶ　ＰＯＲＴＢ，＃０１Ｈ」であると解読する。

【０１３１】ステップＳ１０６で、ＣＰＵモデル４は、解読した命令が周辺回路の読み書きに関するものかを判断する。今の場合、Ｉ／ＯポートＢモデルへのストア命令であるので、「Ｙ」に進む。

【０１３２】ステップＳ１０７で、ＣＰＵモデル４は、解読した命令に対応する機能関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ）を生成し、この機能関数を機能関数Ｉ／Ｆモデル５に渡す。

【０１３３】ステップＳ１０８で、ＣＰＵモデル４は、周辺回路処理サブルーチンの呼出しを実行する。

【０１３４】図１０のステップＳ２０１で、機能関数Ｉ／Ｆモデル５は、仮想モデル７に引数の条件を満たす関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ）が登録されているかどうかのチェックを行う。この関数は図６の機能関数選択テーブル３に登録されていないので、機能関数Ｉ／Ｆモデル５は、機能関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ）をバスＩ／Ｆモデル６に渡し、「Ｎ」に進む。

【０１３５】ステップＳ２０３で、バスＩ／Ｆモデル６は、バスＩ／Ｆモデルシミュレーションを開始する。

【０１３６】ステップＳ２０４で、バスＩ／Ｆモデル６は、受領した機能関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ）をもとに、端子信号コード（図１２）に変換し、システムＩ／Ｆモデル８へ出力する準備を行う。ここで、バスＩ／Ｆモデル６は、第１～第４マシンサイクルＭ１～Ｍ４に対応する／ＣＳ（チップ選択端子信号），ＡＤＲＳ（アドレス端子信号），／ＲＤ（リード端子信号），／ＷＲ（ライト端子信号）の各信号を生成する。なお、「／」は信号がローレベル“０”のとき有効であることを意味する。

【０１３７】ステップＳ２０５で、バスＩ／Ｆモデル６は、第１マシンサイクルＭ１の端子信号を出力する。すなわち、図１２に示すように、マシンサイクルＭ１の時点での各端子信号は、次の状態である（図１２）。

【０１３８】・Ｉ／ＯポートＢモデル１４用のチップ選択端子信号／ＣＳ＝０

・アドレス端子信号ＡＤＲＳ＝Ｉ／ＯポートＢモデル１４のアドレス

・データ端子信号ＤＡＴＡ＝不定

・リード端子信号／ＲＤ＝１

・ライト端子信号／ＷＲ＝１

ステップＳ２１０で、バスＩ／Ｆモデル６は、マイコンシミュレータ１内のＣＰＵモデル４用の端子信号コード（／ＣＳ，ＡＤＲＳ等）から、ハードウェアシミュレータ２用の端子信号コード（不図示）に変換し、システムＩ／Ｆモデル８に各端子信号を渡す。これにより、ハードウェアシミュレータ２側の処理に移行する。

【０１３９】ステップＳ２１３で、システムＩ／Ｆモデル８は前記端子信号のいずれか１つの信号が変化したかを常に監視しており、端子信号のいずれかが変化したことを検出すると、入力データの準備が完了したと判断し、「Ｙ」へ進む。

【０１４０】ステップＳ２１４で、システムＩ／Ｆモデル８は、／ＣＳ，ＡＤＲＳ，／ＲＤ，／ＷＲ，ＤＡＴＡの各端子信号を受け取る。

【０１４１】ステップＳ２１５で、システムＩ／Ｆモデル８は、所定のマシンサイクル（４回）に達しているか否かを判断する。この時点では４回に達していないので、「Ｎ」へ進む。

【０１４２】ステップＳ２１２で、システムＩ／Ｆモデル８は、マシンサイクルＭ１の情報を正確に受信したことをバスＩ／Ｆモデル６に知らせるために、「ＡＣＫ」（受信確認）をバスＩ／Ｆモデル６に渡す。正確に受信できなかった場合には、「ＮＡＣＫ」（受信不能）を返し、バスＩ／Ｆモデル６に再送を促す。

【０１４３】ステップＳ２１０で、システムＩ／Ｆモデル８は、「ＡＣＫ」データを変換して、バスＩ／Ｆモデル６に渡す。これにより、再びマイコンシミュレータ１側の処理に移る。

【０１４４】ステップＳ２０６で、バスＩ／Ｆモデル６は、端子信号のいずれかが変化したことを検出すると、入力データの準備が完了したと判断し、「Ｙ」へ進む。

【０１４５】ステップＳ２０７で、バスＩ／Ｆモデル６は、「ＡＣＫ」を受け取る。

【０１４６】ステップＳ２０８で、バスＩ／Ｆモデル６は、「ＡＣＫ」受信なので、前端子信号は送信完了と判断する。現マシンサイクルは「Ｍ１」なので、通信は完了していないと判断し、「Ｎ」へ進む。

【０１４７】ステップＳ２０５で、バスＩ／Ｆモデル６はシステムＩ／Ｆモデル８にマシンサイクルＭ２の状態を渡す。マシンサイクルＭ２の時点での各端子信号は、次の状態である（図１２）。

【０１４８】・Ｉ／ＯポートＢモデル１４用のチップ選択端子信号／ＣＳ＝０

・アドレス端子信号ＡＤＲＳ＝Ｉ／ＯポートＢのアドレス

・データ端子信号ＤＡＴＡ＝＃０１Ｈ

・リード端子信号／ＲＤ＝１

・ライト端子信号／ＷＲ＝０（書き込みデータが有効であることを示す）

以下、ステップＳ２１０→Ｓ２１３→Ｓ２１４→Ｓ２１５→Ｓ２１２→Ｓ２１０→Ｓ２０６→Ｓ２０７→Ｓ２０８→Ｓ２０５の処理を、ステップＳ２１５においてマシンサイクルＭ４終了を検出するまで繰り返す。

【０１４９】ステップＳ２１５で、システムＩ／Ｆモデル８は、規定のマシンサイクルが完了したことを確認すると、受信した端子信号コードをハードウェアモデル２

０に渡し、「Ｙ」へ進む。

【０１５０】ステップＳ２０９で、ハードウェアモデルシミュレーションを実行する。すなわち、図１のＩ／ＯポートＢモデル１４は、接続情報に基づき演算を行い、出力ポートに出力する値として（ＰＢ３，ＰＢ２，ＰＢ１，ＰＢ０）＝（０，０，０，１）を得る。このうち（ＰＢ１，ＰＢ０）＝（０，１）をデコーダモデル１５に渡し、デコーダモデル１５は、これをデコードして（Ｄ３，Ｄ２，Ｄ１，Ｄ０）＝（０，０，１，０）をキーマトリクスモデル１３に出力するとともに、キー入力ＧＵＩ１８１に渡す。

【０１５１】ステップＳ２１２で、ハードウェアモデル２０内のＩ／ＯポートＢモデル１４は、上記シミュレーションの完了を知らせるため、シミュレーション完了信号「ＤＯＮＥ」をシステムＩ／Ｆモデル８へ渡す。なお、シミュレーションでエラーが発生した場合には、シミュレーションエラー信号「ＥＲＲ」を返す。

【０１５２】ステップＳ２１０で、システムＩ／Ｆモデル８は「ＤＯＮＥ」をデータ変換する。

【０１５３】ステップＳ２０６で、バスＩ／Ｆモデル６は、端子信号の変化を検出し、「Ｙ」へ進む。

【０１５４】ステップＳ２０７で、バスＩ／Ｆモデル６は、「ＤＯＮＥ」を受信する。

【０１５５】ステップＳ２０８で、バスＩ／Ｆモデル６は、「ＤＯＮＥ」受信により、バスＩ／Ｆが完了したと判断し、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に、「ＤＯＮＥ」を返し、「Ｙ」に進む。

【０１５６】以上により、図９に戻り、ステップＳ１０８が終了する。

【０１５７】図９のステップＳ１０９で、ＣＰＵモデル４は、機能関数Ｉ／Ｆモデル５より「ＤＯＮＥ」を受領することで、ストア命令の実行が終了したと検知する。

【０１５８】ステップＳ１１０で、停止要求ではないので、「Ｎ」へ進み、ステップＳ１０１に移る。

【０１５９】《ＲＯＭアドレス１００２Ｈの処理》ステップＳ１０１で、ＣＰＵモデル４は、プログラムカウンタの値「１００２Ｈ」を取得する。以下、１行目と同様に、ステップＳ１０２→Ｓ１０３→２０１→Ｓ２０２→Ｓ３０１→Ｓ３０２→Ｓ３０７と進む。

【０１６０】図９のステップＳ１０４で、ＣＰＵモデル４は、戻り値「Ｃ５０１Ｈ」（ＭＯＶ　Ａ，ＰＯＲＴ　Ａ）を読み込む。

【０１６１】ステップＳ１０５で、命令を解析し、ステップＳ１０６で、「Ｙ」へ進む。

【０１６２】ステップＳ１０７で、ＣＰＵモデル４は、解読した命令に対応する機能関数ＩＯＲＤ（ＰＯＲＴ　Ａ、Ｉ／Ｏ、ＲＤ）を生成し、機能関数Ｉ／Ｆモデル５に渡す。

【０１６３】ステップＳ１０８で、周辺回路処理サブルーチン呼出し図１０のステップＳ２０１で、この機能関数ＩＯＲＤ（ＰＯＲＴＡ）は図６の機能関数選択テーブル３に登録されていないので「Ｎ」へ進む。

【０１６４】ステップＳ２０４で、バスＩ／Ｆモデル６は、各端子信号を出力する準備をする。

【０１６５】ステップＳ２０５～Ｓ２０８で、前記動作を繰り返す。

【０１６６】ステップＳ２０９のシミュレーションの途中で、シミュレータのオペレータが、図２に示す入力手段により画面に表示されたキー入力ＧＵＩ１８２のキーボードの右から２列目の上から３番目（図３）をクリックすることで、キー入力ＧＵＩ１８２はその情報をＩ／ＯポートＡモデル１２の入力端子ＰＡ３～ＰＡ０に伝える。この結果、Ｉ／ＯポートＡモデル１２は、（ＰＡ３，ＰＡ２，ＰＡ１，ＰＡ０）＝（０，１，０，０）を得る。

【０１６７】ステップＳ２１２で、Ｉ／ＯポートＡモデル１２は、シミュレーション結果として「＃０４Ｈ」をシステムＩ／Ｆモデル８に渡す。

【０１６８】ステップＳ２１０で、システムＩ／Ｆモデル８は「＃０４Ｈ」をデータ変換する。

【０１６９】ステップＳ２０６で、「Ｙ」へ進む。

【０１７０】ステップＳ２０７で、バスＩ／Ｆモデル６は、「＃０４Ｈ」を受領し、機能関数Ｉ／Ｆモデル５にこれを渡す。

【０１７１】ステップＳ２０８で、機能関数Ｉ／Ｆモデル５は、「＃０４Ｈ」をＣＰＵモデル４に渡し、サブルーチン処理を終了し、図９のステップ１０８に戻る。

【０１７２】図９のステップＳ１０９で、ＣＰＵモデル４は、「＃０４Ｈ」をレジスタＡに格納する。

【０１７３】ステップＳ１１０で、「Ｎ」に進み、ステップＳ１０１に戻る。

【０１７４】《ＲＯＭアドレス１００３Ｈの処理》上記と同様、ステップＳ１０１→Ｓ１０２→Ｓ１０３→Ｓ２０１→Ｓ２０２→Ｓ３０１→Ｓ３０２→Ｓ３０７→Ｓ１０４→Ｓ１０５と進む。

【０１７５】ステップＳ１０６で、解読結果「ＳＵＢ　Ａ　＃０４Ｈ」は、周辺回路読み書きに関係ないので、機能関数の呼び出しは行われず、「Ｎ」へ。

【０１７６】ステップＳ１０９で、ＣＰＵモデル４内で減算命令「（レジスタＡ）－０４Ｈ」を実行し、処理結果をレジスタＡに「０Ｈ」を格納する。

【０１７７】ステップＳ１１０で、「Ｎ」に進み、ステップＳ１０１に戻る。

【０１７８】《ＲＯＭアドレス１００４Ｈの処理》３行目と同様に、ステップＳ１０１→Ｓ１０２→Ｓ１０３→Ｓ２０１→Ｓ２０２→Ｓ３０１→Ｓ３０２→Ｓ３０７→Ｓ１０４→Ｓ１０５と進む。

【０１７９】ステップＳ１０６で、解読結果「ＢＺ　！１００６Ｈ」は、周辺回路読み書きに関係ないので、「Ｎ」へ移る。

【０１８０】ステップＳ１０９で、ＣＰＵモデル４は、分岐命令「ＢＺ　！１００６Ｈ」を実行する。ここでは、条件（レジスタＡ－０４Ｈ＝０）が成立するので、１００６Ｈ番地へジャンプする。すなわち、ＣＰＵモデル４は、プログラムカウンタＰＣに１００６Ｈをセットする。もし、キー入力がないなど条件が成立しないときは、１００５Ｈ番地の処理が実行されて、１００２Ｈ番地へジャンプする。

【０１８１】《ＲＯＭアドレス１００６Ｈの処理》ステップＳ１０１で、ＣＰＵモデル４は、プログラムカウンタの値「１００６Ｈ」を取得し、ステップＳ１０２→Ｓ１０３と進む。

【０１８２】ステップＳ１０４で、前述と同様な動作で、ＣＰＵモデル４は、戻り値：５ＡＢ５（ＭＯＶ　Ａ，ＲＥＡＤＴＩＭＥ）を取り込む。

【０１８３】ステップＳ１０５で、ＣＰＵモデル４は、戻り値「５ＡＢ５Ｈ」を解析する。

【０１８４】ステップＳ１０６で、周辺回路に関係のある命令なので、「Ｙ」へ進む。

【０１８５】ステップＳ１０７で、ＣＰＵモデル４は、機能関数ＩＯＲＤ（ＲＥＡＤＴＩＭＥ）を生成し、機能関数Ｉ／Ｆモデル５に渡す。なお、ＲＥＡＤＴＩＭＥは、Ｉ／Ｏアドレスを示すラベルである。

【０１８６】ステップＳ１０８で、図１０のステップＳ２０１に移る。ステップＳ２０１で、機能関数Ｉ／Ｆモデル５は、機能関数ＩＯＲＤ（ＲＥＡＤＴＩＭＥ）が機能関数選択テーブル３（図６）の判定条件を満たすので、「Ｙ」に進む。

【０１８７】ステップＳ２０２で、仮想モデルシミュレーションを実行する。図１１に移る。

【０１８８】図１１のステップＳ３０１で、仮想モデル７は、仮想関数変換テーブル（図７）により、ＩＯＲＤ（ＲＥＡＤＴＩＭＥ）→Ｒ＿ＴＭに変換し、仮想時計モデルＶ７２（図４）に渡す。

【０１８９】ステップＳ３０２で、仮想関数Ｒ＿ＴＭはＭＥＭ関数でないため「Ｎ」へ進む。

【０１９０】ステップＳ３０３で、「Ｙ」へ進む。

【０１９１】ステップＳ３０８で、仮想モデル７は、仮想時計モデルＶ７２（図４）より時刻情報を読み出し、機能関数Ｉ／Ｆモデル５に時刻情報「＃１０Ｈ」を渡し、機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４に時刻情報「＃１０Ｈ」を返す。図１０のステップＳ２０２に戻り、さらに、図９のステップＳ１０８に戻る。

【０１９２】図９のステップＳ１０９で、ＣＰＵモデル４は、時刻情報「＃１０Ｈ」をレジスタＡに格納する。

【０１９３】ステップＳ１１０で、「Ｎ」に進み、ステップＳ１０１に戻る。

【０１９４】《ＲＯＭアドレス１００７Ｈの処理》３行目および４行目と同様に、ＣＰＵモデル４内部で処理は完了する命令であるから、ステップＳ１０１→Ｓ１０２→Ｓ１０３→Ｓ２０１→Ｓ２０２→Ｓ３０１→Ｓ３０２→Ｓ３０７→Ｓ１０４→Ｓ１０５→Ｓ１０６→Ｓ１０９と進み、レジスタＡに「０」を格納する。その後、ステップＳ１１０→Ｓ１０１に戻る。

【０１９５】《ＲＯＭアドレス１００８Ｈの処理》３行目，４行目および７行目と同様に、ＣＰＵモデル４内部で処理は完了する命令であるから、ステップＳ１０１→Ｓ１０２→Ｓ１０３→Ｓ２０１→Ｓ２０２→Ｓ３０１→Ｓ３０２→Ｓ３０７→Ｓ１０４→Ｓ１０５→Ｓ１０６→Ｓ１０９と進み、ステップＳ１０９で、分岐命令を実行し、「１００ＡＨ」へジャンプする。

【０１９６】《ＲＯＭアドレス１００ＡＨの処理》ステップＳ１０１で、ＰＣの値「１００ＡＨ」を取得し、ステップＳ１０２→Ｓ１０３→Ｓ１０４→Ｓ１０５→Ｓ１０６→Ｓ１０７と進み、ＣＰＵモデル４は、機能関数ＩＯＷＲ（ＰＯＲＴＣ，＃０１Ｈ）を生成する。

【０１９７】次に、ステップＳ１０８と進み、図１０に移り、ステップＳ２０１→Ｓ２０２と進み、図１１のステップＳ３０１で、仮想モデル７は、機能関数ＩＯＷＲ（ＰＯＲＴＣ，＃０１Ｈ）を仮想関数ＬＭＰ０１に変換する。

【０１９８】ステップＳ３０２→Ｓ３０８と移り、ステップＳ３０８で、仮想モデル７は、仮想関数変換テーブル（図８）をもとに仮想関数ＬＭＰ０１を「１」に変換して、仮想ＬＥＤ表示モデルＶ７４（図４）に渡す。仮想ＬＥＤ表示モデルＶ７４は、ＬＥＤ表示ＧＵＩ１８２に渡し、表示装置（図２）にランプの点灯画像を表示させる。

【０１９９】図１１の処理を終了し、図１０ステップＳ２０２に戻り、さらに図９のステップＳ１０８に戻る。

【０２００】ステップＳ１０９→Ｓ１１０と進み、ステップＳ１０１に戻る。

【０２０１】《ＲＯＭアドレス１００ＢＨの処理》ステップＳ１０１で、ＰＣの値「１００ＢＨ」を取得し、ステップＳ１０２→Ｓ１０３→Ｓ２０１→Ｓ２０２→Ｓ３０１→Ｓ３０２→Ｓ３０７と進み、戻り値「５Ａ０３Ｈ」（ＣＡＬＬ　ＫＥＹＩＮ）を得て、ステップＳ１０３に戻る。

【０２０２】ステップＳ１０４で、ＣＰＵモデル４は、戻り値「５Ａ０３Ｈ」（ＣＡＬＬＫＥＹＩＮ）を読み込み、ステップＳ１０５→Ｓ１０６→Ｓ１０９と進む。

【０２０３】ステップＳ１０９で、ＣＰＵモデル４は、レジスタＡ他の内容をスタックに待避し、プログラムカウンタＰＣにサブルーチンの先頭アドレス３００１Ｈ（ＫＥＹＩＮ）を設定して、ステップＳ１１０を通ってＳ１０１に戻る。

【０２０４】《ＲＯＭアドレス３００１Ｈの処理》ステップＳ１０１で、ＰＣの値「３００１Ｈ」を取得し、ステップＳ１０２→Ｓ１０３→Ｓ１０４と進む。

【０２０５】ステップＳ１０４（命令読込）で、通常の

キースキャン・サブルーチン処理であれば、命令コード「Ａ２０１」（図５）を戻り値として読み込む。しかし、ここでは、図８のアドレス３００１Ｈ～３０１ＦＨに示すようなキースキャン・サブルーチン処理を仮想モデルで置き換えて処理するため、予め、アドレス３００１Ｈの内容をダミーの入出力命令「ＣＡＬＬ（ＫＥＹＩＮ）」に置き換えてあるものとする。あるいは、ラベルＫＥＹＩＮの示すアドレスを他のアドレスに置き換えて、そのアドレスにダミーの入出力命令「ＣＡＬＬ（ＫＥＹＩＮ）」を書いておいてもよい。

【０２０６】ステップＳ１０５→Ｓ１０６→Ｓ１０７と進む。

【０２０７】ステップＳ１０７（機能関数生成）で、ＣＰＵモデル４は、機能関数ＣＡＬＬ（ＫＥＹＩＮ）を生成し、これを機能関数Ｉ／Ｆモデル５に渡す。

【０２０８】ステップＳ１０８で、周辺回路処理サブルーチン呼出しステップＳ２０１で、機能関数Ｉ／Ｆモデル５は、機能関数選択テーブル３と比較。登録されており、かつ時刻が「１０Ｈ」であり、引数の条件を満たすので、「Ｙ」に進む。

【０２０９】ステップＳ２０２で、仮想モデルシミュレーションを実行する。

【０２１０】ステップＳ３０１で、機能関数Ｉ／Ｆモデル５は、仮想関数変換テーブル（図７）により、機能関数ＣＡＬＬ（ＫＥＹＩＮ）を仮想関数「Ｋ＿ＩＮ」に変換する。

【０２１１】ステップＳ３０２で、「Ｎ」へ
ステップＳ３０３で、「Ｎ」へ
ステップＳ３０４で、「Ｙ」へ
ステップＳ３０９で、仮想モデル７は、仮想キー入力モデルＶ７３を起動し、キー入力処理を実行。キー入力ＧＵＩ１８１により、あるキーが押下されると、キーの文字情報を取得。仮想モデル７は、キーの文字情報「＃ＥＣＨ」を機能関数Ｉ／Ｆモデル５に渡す。

【０２１２】ステップＳ１０９で、機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４に戻り値「＃ＥＣＨ」を返す。ＣＰＵモデル４は、戻り値「＃ＥＣＨ」をレジスタＡに格納する。

【０２１３】《ＲＯＭアドレス１００ＣＨの処理》３行目，４行目，７行目および８行目と同様に、ステップＳ１０９で、ＣＰＵモデル４は、レジスタＡの値から「ＥＣＨ」を減算し、演算結果「０Ｈ」をレジスタＡに格納する処理を実行する。ＣＰＵモデル４内で処理は完了し、ステップＳ１０１に戻る。

【０２１４】《ＲＯＭアドレス１００ＤＨの処理》３行目，４行目，７行目，８行目，１２行目および１３行目と同様に、マイコンシミュレータ１内部で処理は完了し、「１００ＦＨ」へジャンプする。

【０２１５】《ＲＯＭアドレス１００ＦＨの処理》ステップＳ１０１で、ＰＣの値「１００ＦＨ」を取得し、Ｓ１０２→Ｓ１０３→Ｓ２０１→Ｓ２０２と進み、戻り値「Ｃ７００Ｈ」（ＭＯＶ　ＰＯＲＴＣ，＃００Ｈ）を取得する。

【０２１６】ステップＳ１０５→Ｓ１０６→Ｓ１０７と進み、機能関数ＩＯＷＲ（ＰＯＲＴＣ，＃００Ｈ）生成する。

【０２１７】ステップＳ１０８→Ｓ２０１→Ｓ２０２と進み、ステップＳ３０１で、仮想モデル７は、機能関数ＩＯＷＲ（ＰＯＲＴＣ，＃００Ｈ）を仮想関数ＬＭＰ００に変換する。

【０２１８】ステップＳ３０２→Ｓ３０３→Ｓ３０４→Ｓ３０５→Ｓ３１０と進む。

【０２１９】ステップＳ３１０で、仮想モデル７内の仮想ＬＥＤ表示モデルＶ７４は、仮想関数ＬＭＰ００をもとに、ＬＥＤ表示ＧＵＩ１８２に「０」を出力して図２の表示装置に消灯画像を表示させる。

【０２２０】以上で、図３にその具体例を示したハードウェアモデル２０についての第１実施例の説明を終えるが、図１に示した第１実施例は、図３の電子手帳にとどまらず、マイコンを使用するなら、その他の電子機器についても適用できるのは言うまでもない。

【０２２１】本実施例の構成を採用することにより、以下の効果が得られる。

【０２２２】第１に、シミュレーションに要する時間を短縮できる。これは、プログラムメモリモデルやデータメモリモデルだけでなく、メモリ以外の周辺回路まで仮想モデルを使用して入出力処理を実行するため、ハードウェアシミュレータ２を使用して周辺回路との読み書きする場合に比べ、シミュレーション結果を取得するまでのでの時間を大幅に短縮できる。例えば、ハードウェアモデルシミュレーションで１マシンサイクルに１０ｍｓかかるとすると、４マシンサイクルには４０ｍｓかかる。これに対して、仮想モデルでは、数μｓで同等の結果が得られるので、ロード／ストア命令をシミュレーションする時間を約１０，０００分の１に低減できる。

【０２２３】第２に、仮想モデル７を使用できる周辺回路を予め機能関数選択テーブル３に登録しておくことにより、仮想モデルシミュレータ７かハードウェアシミュレータ２のどちらのモデルを使うかが即座に選択できる。従って、シミュレーションの初期段階で、ハードウェアモデルの接続や機能が正しく動作することが確認できれば、機能関数選択テーブル３の内容を書き換えるだけで、仮想モデルを使用して検証ができるようになる。選択できるアドレスや時間などいろいろな条件を設定できるので、用途に応じて仮想モデルを使い分けしやすくなる。

【０２２４】第３に、機能関数選択テーブル３には他に影響を及ぼさない周辺回路モデルのみ登録することとしたため、仮想モデル７によるシミュレーション結果が他のハードウェアシミュレーションに影響を及ぼすことが

ない。このため、仮想モデルシミュレータ７を使っても、ハードウェアシミュレータ２を使っても、同様のシミュレーション結果を得ることができる。

【０２２５】第４に、ターゲットプログラム内のサブルーチン処理を１つの仮想モデルで置き換えて検証できるようにしたので、一層処理時間を短縮することができる。本実施例では、キー入力サブルーチンを仮想キー入力モデルに置き換える例を示したが、これに限定されるものではなく、液晶表示用ドライバ、警告音出力回路、あるいは、通信装置などを制御するサブルーチンも仮想モデルに置き換えることができる。例えば、液晶表示用ドライバを使って文字を表示するためには、文字コードを文字パターンに変換し、Ｘ方向のドライバとＹ方向のドライバにそれぞれ文字パターンデータを所定のタイミングで供給し、液晶表示装置上に表示させることになる。これをハードウェアシミュレータ２を使って検証しようとすると、非常に複雑で、長いステップのターゲットプログラムを処理することになり、１文字をキー入力モデルから入力して、この文字データを液晶表示装置に表示させるだけで数分を要してしまう。この機能を液晶表示装置と等価な表示ができる仮想モデルで代替えすることで、実使用状況と同程度の速度で表示装置に表示させることができる。

【０２２６】［第２実施例］第１実施例は、ハードウェアシミュレーションによって得られる結果と等価な結果になるような機能を予め仮想モデルに登録しておき、ロード／ストア命令を実行するときに、仮想モデルを使用することにより、システムシミュレーションの高速化を図ったものであった。したがって、仮想モデルに登録するモデルの種類が多いほど望ましい。しかしながら、１つの周辺回路のシミュレーション結果が他の周辺回路のシミュレーションに影響を及ぼす場合には、ハードウェアシミュレータで検証した後で、仮想モデルで検証したり、逆に、仮想モデルで検証した後で、ハードウェアシミュレーションを実行すると、その結果が矛盾することになることがある。

【０２２７】例えば、出力ポートと入力ポートとの間にスイッチが接続されているとする。出力ポートの値を「１」として、入力ポートの値を読み取ることで、スイッチがＯＮしているか、ＯＦＦしているかが分かる。即ち、入力ポートの値が「１」であればスイッチはＯＮしており、「０」であればＯＦＦしていることが分かる。

【０２２８】いま、仮想モデルを使用して出力ポートの値を「１」として、ハードウェアモデルを使って入力ポートの値を読み取るとする。第１実施例あるいは従来のシステムシミュレータでは、仮想モデルとハードウェアモデルとはなんら関連性がないので、仮想モデルの出力ポートの値を「１」にしたからといって、ハードウェアモデルの出力ポートが「１」になっているとは限らない。仮に、ハードウェアモデルの出力ポートが「０」になっているとすると、たとえスイッチがＯＮしていたとしても、入力ポートの値は「０」であり、スイッチがＯＦＦしていると誤判定されてしまうことになる。

【０２２９】そこで、第１実施例では、上述のような不具合が生じないようにするため、シミュレータ結果が互いに関連するモデルは、機能関数選択テーブル（図６）で選択できないようにしたり、それらをまとめて選択するように登録を工夫している。

【０２３０】第２実施例は、あるモデルのシミュレーション結果が他のモデルのシミュレーションに影響を及ぼすような場合であっても、正しいシミュレーション結果が得られるような構成を採ることにより、システムシミュレーションをより高速化したものである。

【０２３１】図１３は、本発明のシステムシミュレータの第２実施例の機能ブロック図である。図１３を図１と対比すれば、第２実施例は第１実施例にデータ更新Ｉ／Ｆモデル９を付加した点が異なっているのがわかる。データ更新Ｉ／Ｆモデル９は、仮想モデル７１のシミュレーション結果をハードウェアモデル２１に反映し、またハードウェアモデル２１のシミュレーション結果を仮想モデル７１に反映させる役割を担う。このような更新機能を設けることにより、１つの周辺回路のシミュレーション結果が他の周辺回路のシミュレーションに影響を及ぼす場合であっても、仮想モデルミュレーションとハードウェアシミュレーションとの間の矛盾を回避することができるのである。

【０２３２】第２実施例についても、図２に示したシミュレーション装置のブロック図およびその説明は第１実施例におけるのと同様に適用できる。したがって、第２実施例のシステムシミュレータによるシステムシミュレーションを実行するときには、上記シミュレーション装置の準備と、マイコンシミュレータ１－１およびハードウェアシミュレータ２－１のロードが前提となる。

【０２３３】図１４は、第２実施例におけるハードウェアモデル２１の構成図であり、ＨＰＭテーブルＨ３０２付きのプログラムメモリモデルＨ３０１と、ＨＤＭテーブルＨ３０４付きのデータメモリモデルＨ３０３と、ＨＩＰ入力レジスタＨ３１０付きの入力ポートモデルＨ３０９と、ＨＫＤ出力レジスタＨ３１３付きの出力ポートモデルＨ３１１およびデコーダモデルＨ３１２とから成る。

【０２３４】プログラムメモリモデルＨ３０１，データメモリモデルＨ３０３，入力ポートモデルＨ３０９，出力ポートモデルＨ３１１は、システムＩ／Ｆモデル８と接続され、ＨＰＭテーブルＨ３０２，ＨＤＭテーブルＨ３０４，ＨＩＰ入力レジスタＨ３１０，ＨＫＤ出力レジスタＨ３１３は、データ更新Ｉ／Ｆモデル９に接続され、それぞれプログラムメモリモデルＨ３０１，データメモリモデルＨ３０３，入力ポートモデルＨ３０９，出力ポートモデルＨ３１１およびデコーダモデルＨ３１２

の更新のために使用されるとともに、仮想モデル７１の更新のために出力される。なお、ＨＰＭテーブルＨ３０２は、プログラムメモリの内容に相当し、ＲＯＭのようにその内容が更新されることがないとわかっている場合には、データ更新Ｉ／Ｆモデル９と接続しなくてもよい。また、入力ポートモデルＨ３０９，出力ポートモデルＨ３１１およびデコーダモデルＨ３１２は、図３におけるＩ／ＯポートＡモデル１２，Ｉ／ＯポートＢモデル１４およびデコーダモデル１５に相当する。

【０２３５】ＨＰＭテーブルＨ３０２は、ターゲットプログラムの命令コードを有しており、図５に示すようなシンボルと戻り値とが対になったデータを記憶手段に記憶している。プログラムメモリモデルＨ３０１は、機能関数Ｉ／Ｆモデル５が出力する機能関数に基づき、所定のアドレスの命令コードをＨＰＭテーブルＨ３０２から読み出して、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に命令コードを返す。

【０２３６】同様に、ＨＤＭテーブルＨ３０４は、データメモリのアクセスアドレスと記憶データとが対になったテーブル（記憶手段）を有しており、データメモリモデルＨ３０３は、機能関数に基づき、所定のアドレスの記憶データをＨＤＭテーブルＨ３０４に書き込んだり、読み出して、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４との間でデータの授受を行う。データメモリモデルＨ３０３は、テーブルの記憶内容が書き換えられると、データ更新Ｉ／Ｆモデル９を介して仮想モデル７１内のＶＤＭテーブルＶ４０４の内容を同一にすべく更新する。

【０２３７】出力ポートモデルＨ３１１は、ＣＰＵモデル４から機能関数Ｉ／Ｆモデル５を介して供給されたポート出力信号をデコーダモデルＨ３１２に出力し、デコーダモデルＨ３１２はポート信号をデコードし、キーマトリクスの列数（または行数）に相当する数のデコード信号を出力する。このデコード信号は、いずれか１つが「１」で他は「０」の信号であって、「１」になる信号の位置を順次変えることで、キースキャンを実行する。ＨＫＤ出力レジスタＨ３１３は、このデコード信号をレジスタ（記憶手段）に保存するとともに、データ更新Ｉ／Ｆモデル９を介して仮想モデル７１内のＶＫＤ出力レジスタＶ４１０の内容を同一にすべく更新する。また、このデコード信号は、キー入力ＧＵＩ１８１にも供給される。

【０２３８】入力ポートモデルＨ３０９は、キー入力ＧＵＩ１８１のキーマトリクスと接続され、キー入力ＧＵＩ１８１から入力されたキー入力信号を読み込み、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に返す。ポート入力は、キーマトリクスの行数（または列数）に相当する数だけのポートを有し、ポート入力処理は、列数（または行数）回の入力処理を繰り返す。ＨＫＤ出力レジスタＨ３１３は、キー入力ＧＵＩ１８１から入力されたキー入力信号を一旦レジスタ（記憶手段）に保存するとともに、データ更新Ｉ／Ｆモデル９を介して仮想モデル７１内のＶＩＰ入力レジスタＶ４０８の内容を同一にすべく更新する。

【０２３９】図１５は、第２実施例における仮想モデル７１の構成図であり、ＶＰＭテーブルＶ４０２付きの仮想プログラムメモリモデルＶ４０１，ＶＤＭテーブルＶ４０４付きの仮想データメモリモデルＶ４０３，ＶＩＰ入力レジスタＶ４０８付きの仮想入力ポートモデルＶ４０７およびＶＫＤ出力レジスタＶ４１０付きの仮想キーデコーダモデルＶ４０９から成る。

【０２４０】仮想プログラムメモリモデルＶ４０１，仮想データメモリモデルＶ４０３，仮想入力ポートモデルＶ４０７，仮想キーデコーダモデルＶ４０９は、機能関数Ｉ／Ｆモデル５と接続され、ＶＰＭテーブルＶ４０２，ＶＤＭテーブルＶ４０４，ＶＩＰ入力レジスタＶ４０８，ＶＫＤ出力レジスタＶ４１０は、データ更新Ｉ／Ｆモデル９に接続され、それぞれ仮想プログラムメモリモデルＶ４０１，仮想データメモリモデルＶ４０３，仮想入力ポートモデルＶ４０７，仮想キーデコーダモデルＶ４０９の更新のために使用されるとともに、ハードウェアモデル２１の更新のために出力される。

【０２４１】図４における仮想キー入力モデルＶ７３は、キー入力ＧＵＩ１８１から、文字情報を得る機能を構成していたが、図１５における仮想入力ポートモデルＶ４０７および仮想キーデコーダモデルＶ４０９は、キーマトリクス部分の入力信号と出力信号のレベル（「ハイレベル」または「ロウレベル」）を取り扱う点において、第１実施例の仮想キー入力モデルＶ７３とは大きく異なる。

【０２４２】ＶＰＭテーブルＶ４０２は、ターゲットプログラムの命令コードを有しており、図５に示すようなシンボルと戻り値とが対になったデータをテーブル（記憶手段）に記憶している。仮想プログラムメモリモデルＶ４０１は、機能関数Ｉ／Ｆモデル５が出力する機能関数に基づき、所定のアドレスの命令コードをＶＰＭテーブルＶ４０２から読み出して、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に命令コードを返す。

【０２４３】同様に、ＶＤＭテーブルＶ４０４は、データメモリのアクセスアドレスと記憶データとが対になったテーブル（記憶手段）を有しており、仮想データメモリモデルＶ４０３は、機能関数に基づき、所定のアドレスの記憶データをＶＤＭテーブルＶ４０４に書き込んだり、読み出して、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４との間でデータの授受を行う。仮想データメモリモデルＶ４０３は、テーブルＶ４０４の記憶内容が書き換えられると、データ更新Ｉ／Ｆモデル９を介してハードウェアモデル２１内のＨＤＭテーブルＨ３０４の内容を同一にすべく更新する。

【０２４４】仮想キーデコーダモデルＶ４０９は、ＣＰ

Ｕモデル４から機能関数Ｉ／Ｆモデル５を介して供給されたポート出力信号をデコーダをデコードし、キーマトリクスの列数（または行数）に相当する数のデコード信号を出力する。このデコード信号は、いずれか１つが「１」で他は「０」の信号であって、「１」になる信号の位置を順次変えることで、キースキャンを実行する。ＶＫＤ出力レジスタＶ４１０は、このデコード信号をレジスタ（記憶手段）に保存するとともに、データ更新Ｉ／Ｆモデル９を介してハードウェアモデル２１内のＨＫＤ出力レジスタＨ３１３の内容を同一にすべく更新する。また、このデコード信号は、キー入力ＧＵＩ１８１にも供給される。

【０２４５】ハードウェアモデル２１は、出力ポートモデルＨ３１１とデコーダモデルＨ３１２との接続情報とこれに入力される端子信号をもとに演算を行い、各内部回路に信号を伝搬させ、出力端子まで到達させることでデコード信号を得る。このため、出力ポートモデルＨ３１１とデコーダモデルＨ３１２とは、デコード信号を得るのに数１０～数１００ｍｓの時間を要する。これに対して、仮想モデル７１は、出力ポートに出力するデータをもとに論理演算するか、変換テーブルによりデコード信号を得るようにしている。仮想モデル７１内の仮想キーデコーダモデルＶ４０９は、例えば、ポート出力データ（０，１）をデコード信号（０，０，１，０）に変換するだけであるので、数μｓ程度の極めて短時間で結果を得ることができる。

【０２４６】仮想入力ポートモデルＶ４０７は、キー入力ＧＵＩ１８１のキーマトリクスと接続され、キー入力ＧＵＩ１８１から入力されたキー入力信号を読み込み、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に返す。ポート入力は、キーマトリクスの行数（または列数）に相当する数だけのポートを有し、ポート入力処理は、列数（または行数）回の入力処理を繰り返す。ＶＩＰ入力レジスタＶ４０８は、キー入力ＧＵＩ１８１から入力されたキー入力信号を一旦レジスタ（記憶手段）に保存するとともに、データ更新Ｉ／Ｆモデル９を介してハードウェアモデル２１内のＨＩＰ入力レジスタＨ３１０の内容を同一にすべく更新する。

【０２４７】仮想入力ポートモデルＶ４０７は、キー入力信号をそのままＣＰＵモデル４に返すだけなので数μｓで処理が終わる。これに対して、ハードウェアモデル２１内の入力ポートモデルＨ３０９は、接続情報とこれに入力される端子信号をもとに演算を行い、各内部回路に信号を伝搬させ、出力端子まで到達させることでキー入力信号をＣＰＵモデル４に返すので、処理が終わるのに数１０ｍｓもかかる。

【０２４８】図１６は、機能関数選択テーブル３１の内容例であり、機能関数Ｉ／Ｆモデル５が、仮想モデル７１を選択可能な機能関数とその判断条件を示す。ここで、ＰＯＲＴＢは、出力ポートモデルＨ３１１のＩ／Ｏアドレスを表すラベルである。これ以外の本テーブル中の引数とその意味は図６についての説明がそのまま当てはまるため説明は省略する。機能関数選択テーブル３１の内容は、ハードウェアシミュレーションの対象を何にするか、何を仮想モデルで置き換えられるかによって決まり、常に固定されているものではない。また、機能関数選択テーブル３１には、仮想モデルに登録されていない機能関数を登録することはできないが、仮想モデルに登録されているモデルに対応する機能関数を全て登録する必要はない。

【０２４９】図１７は、第２実施例における周辺回路処理サブルーチンの詳細フローチャートを示し、図９における周辺回路処理サブルーチンＳ１０３，Ｓ１０８によって呼び出されるものである。同図中、実線は処理の流れ、点線はデータの流れを示す。

【０２５０】第１実施例と同様、ＣＰＵモデル４が、命令コードをフェッチしたり、ロード命令やストア命令など周辺装置との入出力を伴う命令を実行した場合、周辺回路処理サブルーチンが呼び出される。以下、図１３を参照して、図１７の流れを説明する。

【０２５１】図１７において、ＣＰＵモデル４が機能関数Ｉ／Ｆモデル５に機能関数を渡すと、ステップＳ４０１で、機能関数Ｉ／Ｆモデル５は、機能関数選択テーブル３１を参照して、仮想モデルシミュレーション（Ｓ４０２）を使うか、ハードウェアシミュレーション（Ｓ４０９）を使うか判断する。機能関数Ｉ／Ｆモデル５は、ＣＰＵモデル４から受け取った機能関数が機能関数選択テーブル３１に登録されていて、かつその判断条件を満たしていれば、「Ｙ」へ進み、仮想モデルシミュレーション（Ｓ４０２）を実行し、機能関数が登録されていなかっり、または判断条件を満たしていなければ、「Ｎ」へ進み、バスＩ／Ｆモデルミュレーション（Ｓ４０３）に移る。

【０２５２】ステップＳ４０２（仮想モデルシミュレーション）で、仮想モデル７１は、後述の仮想モデルシミュレーションを実行する。実行終了後、ミュレーション結果を各モデルに付随するファイルやレジスタなどの記憶手段に保存するとともに、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４にその結果を返す。

【０２５３】ステップＳ４２０（ハードウェアデータ更新起動）で、仮想モデル７１は、仮想モデルシミュレーションを実行後、必要に応じてハードウェアデータ更新起動を行い、ハードウェアモデル２１に必要なデータを供給する。この結果、ハードウェアモデル２１は、所定のファイルやレジスタのデータ更新を行い（Ｓ４２３）、仮想モデルシミュレーションの結果をハードウェアモデル２１に反映する。ここで、プログラムメモリモデルＨ３０１のように、データ更新する必要がないものについては、初めから本ステップをスキップするようにしてもよい。

【０２５４】一方、ステップＳ４０１において、「Ｎ」に進み、ハードウェアシミュレータ２－１を使う場合には、ステップ４０３に移る。ステップＳ４０３（バスＩ／Ｆモデルシミュレーション）で、バスＩ／Ｆモデル６は、機能関数Ｉ／Ｆモデル５から機能関数を受領し、ハードウェアモデルシミュレーションに必要な端子信号に変換する。なお、バスＩ／Ｆモデルシミュレーション Ｓ４０３の詳細は図１０におけるバスＩ／Ｆモデルシミュレーション Ｓ２０３に示すとおりである。また、端子信号については、第１実施例の図１２と同じである。

【０２５５】ステップＳ４１０（シミュレータ間Ｉ／Ｆ処理）で、バスＩ／Ｆモデル６は、前記端子信号名や数値、コードなどをハードウェアシミュレータ２－１で使用する信号端子名や、数値、コードなどに変換したり、信号授受のタイミングを調整する。なお、マイコンシミュレータ１－１とハードウェアシミュレータ２－１で信号端子名などのコード体系が同一である場合には、本ステップをスキップするようにしてもよい。

【０２５６】ステップＳ４１１（システムＩ／Ｆモデルシミュレーション）で、システムＩ／Ｆモデル８は、バスＩ／Ｆモデル６より、前記端子信号を受領し、ハードウェアモデル２１に渡す。なお、システムＩ／Ｆモデルシミュレーション Ｓ４１１の詳細は図１０におけるシステムＩ／Ｆモデルシミュレーション Ｓ２１１に示すとおりである。

【０２５７】ステップＳ４０９（ハードウェアシミュレーション）で、ハードウェアモデル２１は、入力された端子信号とハードウェアモデル２１の接続情報に基づき、モデル内の接続点の状態を演算によって求め、出力点の状態がどのようになるかを求める。ハードウェアモデル２１は、このようにして求めた結果を、システムＩ／Ｆモデル８に返す。

【０２５８】ステップＳ４２４（仮想モデル書き替え必要？）で、ハードウェアモデル２１は、機能関数選択テーブル３１を参照して、仮想モデル７１に付随する所定のファイルやレジスタのデータ更新を行う必要があるか否かを判断する。Ｓ４０９で処理したハードウェアシミュレーションが、機能関数選択テーブル３１に登録されていて、かつ引数が判断条件を満たす場合には、ハードウェアモデル２１は、「Ｙ」へ進み、ステップＳ４２５の処理に移る。逆に、Ｓ４０９で処理したハードウェアシミュレーションが、機能関数選択テーブル３１に登録されていないか、または引数が判断条件を満たさない場合には、ハードウェアモデル２１は、「Ｎ」へ進み、ステップＳ４１１の処理に移る。

【０２５９】仮想モデル７１に登録されているサブモデルは、ハードウェアモデル２１に必ず存在するので、データ更新起動（Ｓ４２０）を処理するとき、書き替えの必要性を判断する必要はなかった。しかし、ハードウェアモデル２１に登録されているサブモデルは、仮想モデル７１に必ずしも存在するとは限らないので、データ更新起動（Ｓ４２５）を処理する前に、書き替えの必要性を判断する処理（Ｓ４２４）が必要となる。

【０２６０】ステップＳ４２５（仮想モデルデータ更新起動）で、ハードウェアモデル２１は、ハードウェアモデルシミュレーション（Ｓ４０９）の結果を仮想モデル７１に渡し、仮想モデル７１は、所定のファイルやレジスタのデータ更新を行い、前記結果を反映する。

【０２６１】ステップＳ４１１（システムＩ／Ｆモデルシミュレーション）で、システムＩ／Ｆモデル８は、ハードウェアシミュレーション（Ｓ４０９）の結果をステップＳ４１０の処理を介してバスＩ／Ｆモデル６に返す。

【０２６２】ステップＳ４０３（バスＩ／Ｆモデルシミュレーション）で、バスＩ／Ｆモデル６は、ハードウェアシミュレーション（Ｓ４０９）の結果を機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に渡す。

【０２６３】以上の処理が終了すると、本サブルーチンを呼び出したステップＳ１０３またはＳ１０８（図９）に戻る。

【０２６４】図１８は、第２実施例における仮想モデルシミュレーション Ｓ４０２（図１７）の詳細フローチャートを示す。

【０２６５】図１８において、ステップＳ６０１で、仮想モデル７１は、機能関数Ｉ／Ｆモデル５から機能関数を受け取ると、図７と同様、機能関数を仮想モデル７１のコード体系にあうように仮想関数に変換する。例えば、命令フェッチ関数ＦＥＴＣＨ（ｍ）はＭＥＭｍに、データメモリ読み書き関数ＭＥＭＲＷ（ｍ）はＤＭＥに、ポートＡ入力関数ＩＯＲＤ（ＰＯＲＴＡ）はＩＰＴａに、ポートＢ出力関数ＩＯＷＲ（ＰＯＲＴＢ、ｎ）はＫＤＣｎにそれぞれ変換される。なお、コード体系が同じであれば、本ステップは省略してもよい。

【０２６６】ステップＳ６０２で、仮想モデル７１は、仮想関数がＦＥＴＣＨ関数ＭＥＭｍであるかを判断し、ＭＥＭｍであれば、「Ｙ」へ進み、ステップＳ６０３へ移る。ＭＥＭｍでなければ、「Ｎ」へ進み、ステップＳ６０４へ移る。

【０２６７】ステップＳ６０４で、仮想モデル７１は、仮想関数がデータメモリ読み書き関数ＤＭＥｍであるかを判断し、ＤＭＥｍであれば、「Ｙ」へ進み、ステップＳ６０５へ移る。ＤＭＥｍでなければ、「Ｎ」へ進み、ステップＳ６１０へ移る。

【０２６８】ステップＳ６１０で、仮想モデル７１は、仮想関数がポートＡ入力関数ＩＰＴａであるかを判断し、ＩＰＴａであれば、「Ｙ」へ進み、ステップＳ６１１へ移る。ＩＰＴａでなければ、「Ｎ」へ進み、ステップＳ６１２へ移る。

【０２６９】ステップＳ６１２で、仮想モデル７１は、仮想関数がポートＢ出力関数ＫＤＣｎであるかを判断

し、ＫＤＣｎであれば、「Ｙ」へ進み、ステップＳ６１３へ移り、ＫＤＣｎでなければ、「Ｎ」へ進み、ステップＳ６１４へ移る。

【０２７０】ステップＳ６１４で、仮想モデル７１は、受領した仮想関数が登録されていない関数であったり、パラメータが違っていたりすると、表示装置にエラーメッセージを表示する。

【０２７１】ステップＳ６０３で、仮想モデル７１内の仮想プログラムメモリモデルＶ４０１（図１５）は、テーブルを参照して、ｍ番地の命令コードをＣＰＵモデル４に返す。テーブルは、図５に示すターゲットプログラムのテーブルに相当し、これを予めＶＰＭテーブルＶ４０２に登録してある。仮想プログラムメモリモデルＶ４０１は、仮想関数を示すシンボルＭＥＭｍを受け取ると、このシンボルに対応する戻り値を読み出し、機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に返し、処理を終了する。

【０２７２】通常、ＶＰＭテーブルＶ４０２のプログラム内容は書き替わることがないので、プログラムメモリモデルＨ３０１のＨＰＭテーブルＨ３０２の内容を更新する必要がない。従って、図１７では、ステップＳ６０３終了後、ステップＳ４２０を介さずに仮想モデルシミュレーションＳ４０２の処理を終了するようにしている。

【０２７３】ステップＳ６０５で、仮想モデル７１内の仮想データメモリモデルＶ４０３（図１５）は、ＣＰＵモデル４とデータを授受し、テーブル内の指定の場所にデータを読み書きする。テーブルは、データメモリモデルの記憶容量に相当する行の記憶領域を有し、テーブルの１つの行はアドレス情報を含むシンボル（ＤＭＥｍ）と読み書きデータとで構成される。ＶＤＭテーブルＶ４０４には、データメモリモデルの最下位アドレスｋから最上位アドレス１までのシンボルＤＭＥｋ～ＤＭＥ１が予め付与されている。書き込みの場合、仮想データメモリモデルＶ４０３は、ＣＰＵモデル４から仮想関数「ＤＥＭｍ，ｄ」を受け取ると、このシンボルに対応する位置ＤＭＥｍにデータｄを書き込む。読み出しの場合には、仮想データメモリモデルＶ４０３は、ＣＰＵモデル４からデータ情報を含まない仮想関数「ＤＥＭｍ」を受け取ると、このシンボルに対応する位置ＤＭＥｍからデータｄを読み出し、このデータｄを機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に返す。

【０２７４】ステップＳ６１１で、仮想モデル７１内の仮想入力ポートモデルＶ４０７（図１５）は、キー入力ＧＵＩ１８１から押下されたキーマトリクスの出力レベル情報を取得し、一旦ＶＩＰ入力レジスタＶ４０８に保存するとともに、その出力レベル情報を戻り値として機能関数Ｉ／Ｆモデル５を介してＣＰＵモデル４に返し、処理を終了する。一方、ＶＩＰ入力レジスタＶ４０８の内容は、ステップＳ４２０の処理に渡され、データ更新Ｉ／Ｆモデル９（図１３）はＨＩＰ入力レジスタ（図１４）の内容を更新する。

【０２７５】ステップＳ６１３（データ出力・デコード処理）で、仮想モデル７１内の仮想キーデコーダモデルＶ４０９（図１５）は、受領した機能関数ＩＯＷＲ（ＰＯＲＴＢ、Ｄ＊）をもとに、出力ポートモデルＨ３１１とデコーダモデルＨ３１２（図１５）と等価な端子出力レベルが得られるように処理して、この端子出力をキー入力ＧＵＩ１８１に供給する。例えば、ポート端子（ＰＢ１，ＰＢ０）にデータ（ｄ１，ｄ０）を出力する場合、機能関数は、ＩＯＷＲ（ＰＯＲＴＢ、ｄ１・ｄ０）で表され、仮想キーデコーダモデルＶ４０９は、次の論理演算を行い、端子出力レベルＤ３，Ｄ２，Ｄ１，Ｄ０を得る。

【０２７６】Ｄ３＝ｄ１＊ｄ０
Ｄ２＝ｄ１＊／ｄ０
Ｄ１＝／ｄ１＊ｄ０
Ｄ０＝／ｄ１＊／ｄ０

ここで、「／」は信号ｄ１またはｄ０の論理否定（反転信号）を意味する。

【０２７７】得られた演算結果は、ＶＫＤ出力レジスタＶ４１０に保存されるとともに、ＶＩＰ入力レジスタＶ４０８の内容は、ステップＳ４２０の処理に渡され、データ更新Ｉ／Ｆモデル９（図１３）はＨＫＤ出力レジスタ（図１４）の内容を更新する。

【０２７８】このように、データ出力・デコード処理Ｓ６１３は、ハードウェアモデルの処理とはことなり、前記演算によってキー入力ＧＵＩ１８１のキーマトリクスモデルへ供給する端子出力レベルを求めるので、数μｓの短時間で仮想モデルシミュレーション結果を得ることができる。

【０２７９】ステップＳ４２０（ハードウェアモデルデータ更新起動）で、仮想モデル７１は、各テーブルＶ４０４、または各レジスタＶ４０８，Ｖ４１０の内容を、データ更新Ｉ／Ｆモデル９（図１３）に渡す。データ更新Ｉ／Ｆモデル９は、ステップＳ４２３（図１７）のデータ更新を起動し、ハードウェアモデル２１内の仮想モデルに対応する各テーブルＨ３０４、または各レジスタＨ３１０，Ｈ３１３の内容を更新する。

【０２８０】以上の仮想モデルシミュレーションＳ４０２の処理が終了すると、マイコンシミュレータ１－１は、図１７の処理に戻り、ステップＳ４０２を終了して、更に図９のサブルーチン呼び出し元Ｓ１０３またはＳ１０８へ戻る。

【０２８１】図１９は、図１７のうち１点鎖線で示された部分、即ち、ハードウェアシミュレーションＳ４０９と仮想モデルデータ更新Ｓ４２４，Ｓ４２５の詳細フローチャートを示す。

【０２８２】まず、ハードウェアモデル２１は、システムＩ／Ｆモデル８から端子信号を受け取ると、Ｓ８０１

の処理に移る。

【０２８３】ステップＳ８０１（ＭＥＭ＝０？）で、ハードウェアモデル２１（図１３）は、端子信号ＭＥＭを確認して、プログラムメモリモデルＨ３０１またはデータメモリモデルＨ３０３の呼び出しか否かを判断する。端子信号ＭＥＭが「０」であれば、ハードウェアモデル２１は、「Ｙ」へ進み、ステップＳ８０２の処理に移り、「１」であれば、「Ｎ」へ進み、ステップＳ８０４の処理に移る。

【０２８４】ステップＳ８０２（Ｉ／Ｏ＝０？）で、ハードウェアモデル２１（図１３）は、端子信号Ｉ／Ｏを確認して、メモリを除く周辺回路Ｈ３０９，Ｈ３１１（図１４）の呼び出しか否かを判断する。端子信号Ｉ／Ｏが「０」であれば、ハードウェアモデル２１は、「Ｙ」へ進み、ステップＳ８０７の処理に移り、「１」であれば、「Ｎ」へ進み、ステップＳ８０９の処理に移る。

【０２８５】メモリ関連の処理である場合、ステップＳ８０２の処理に移る。

【０２８６】ステップＳ８０２（０≦ＡＤＲＳ＜２０００？）で、ハードウェアモデル２１（図１３）は、端子信号ＡＤＲＳを確認して、アドレス信号が０００００Ｈ番地以上で２０００Ｈ番地未満かを判断する。端子信号ＡＤＲＳが前記範囲内であれば、ハードウェアモデル２１は、プログラムメモリモデルＨ３０１（図１４）の読込処理であると判断して「Ｙ」へ進み、ステップＳ８１０の処理に移り、範囲外であれば、「Ｎ」へ進み、ステップＳ８０３の処理に移る。

【０２８７】ステップＳ８０３（ＡＤＲＳ≧２０００？）で、ハードウェアモデル２１（図１３）は、端子信号ＡＤＲＳを確認して、アドレス信号が２０００Ｈ番地以上かを判断する。端子信号ＡＤＲＳが前記範囲内であれば、ハードウェアモデル２１は、データメモリモデルＨ３０３（図１４）の読み書き処理であると判断して「Ｙ」へ進み、ステップＳ８１１の処理に移り、範囲外であれば、「Ｎ」へ進み、ステップＳ８０９の処理に移る。

【０２８８】メモリ以外の周辺回路関連の処理である場合、ステップＳ８０７の処理に移る。

【０２８９】ステップＳ８０７（ＩＰＴ？）で、ハードウェアモデル２１（図１３）は、端子信号ＡＤＲＳを確認して、アドレス信号がＰＯＲＴＡ番地か否かを判断する。端子信号ＡＤＲＳがＰＯＲＴＡ番地であれば、ハードウェアモデル２１は、入力ポートモデルＨ３０９（図１４）の読み込み処理であると判断して「Ｙ」へ進み、ステップＳ８１４の処理に移り、ＰＯＲＴＡ以外であれば、「Ｎ」へ進み、ステップＳ８０８の処理に移る。

【０２９０】ステップＳ８０８（ＯＰＴ？）で、ハードウェアモデル２１（図１３）は、端子信号ＡＤＲＳを確認して、アドレス信号がＰＯＲＴＢ番地か否かを判断する。端子信号ＡＤＲＳがＰＯＲＴＢ番地であれば、ハードウェアモデル２１は、出力ポートモデルＨ３０９（図１４）の書き込み処理であると判断して「Ｙ」へ進み、ステップＳ８１５の処理に移り、ＰＯＲＴＢ以外であれば、「Ｎ」へ進み、ステップＳ８０９の処理に移る。

【０２９１】ステップＳ８０１～Ｓ８０４、Ｓ８０７，Ｓ８０８で何れも「Ｎ」であった場合、ステップＳ８０９（エラー処理）で、ハードウェアモデル２１（図１３）は、該当処理が存在しない旨を表示装置（図２）に表示するとともに、システムＩ／Ｆモデル８にエラー信号「ＥＲＲ」を渡して、処理を終了する。

【０２９２】一方、ステップＳ８１０（プログラム読込処理）で、プログラムメモリモデルＨ３０１は、ＨＰＭテーブルＨ３０２（図１４）を参照して、アドレス端子信号ＡＤＲＳに対応する命令コードを読み出し、命令コードをシステムＩ／Ｆモデル８に返す。ここで、プログラムメモリモデルＨ３０１は、接続情報に基づき入力された端子信号を各接続点に伝搬させ、出力端子の信号レベル（命令コード）を演算処理して求める。このため、回路規模が大きいほど、演算結果が出るのに時間がかかる。

【０２９３】プログラムメモリモデルＨ３０１内のＨＰＭテーブルＨ３０２は、シミュレーションの最中に内容が書き変わることがないので、仮想モデル７１のデータ更新をする必要はなく、Ｓ８１０の処理が終了すると、図１７のステップＳ４０９に戻り、Ｓ４１１の処理に移る。

【０２９４】ステップＳ８１１（データメモリ読み書き処理）で、データメモリモデルＨ３０３は、システムＩ／Ｆモデル８との間で授受する記憶データをＨＤＭテーブルＨ３０４（図１４）に読み書きする。書き込みの場合、データメモリモデルＨ３０３は、アドレス端子信号ＡＤＲＳ番地とこれに対応する記憶データＤＡＴＡをシステムＩ／Ｆモデル８から受け取り、ＨＤＭテーブルＨ３０４内の所定の領域に記憶する。読み出しの場合、データメモリモデルＨ３０３は、アドレス端子信号ＡＤＲＳ番地をシステムＩ／Ｆモデル８から受け取り、ＨＤＭテーブルＨ３０４内の所定の領域から記憶データＤＡＴＡを読み出して、システムＩ／Ｆモデル８に返す。ここで、データメモリモデルＨ３０３は、接続情報に基づき入力された端子信号を各接続点に伝搬させ、出力端子の信号レベル（記憶データ）を演算処理して求める。本処理終了後は、ステップＳ４２４に移る。

【０２９５】ステップＳ８１５（データ出力・デコード処理）で、出力ポートモデルＨ３１１（図１４）は、システムＩ／Ｆモデル８から、アドレス信号ＰＯＲＴＢ、データＤＡＴＡなどを受け取り、接続情報に基づいて、ポート出力信号を求める。デコードモデルＨ３１２（図１４）は、ポート出力信号を入力信号として、接続情報に基づいて、デコード出力信号を求める。このデコード

出力信号は、ＨＫＤ出力レジスタＨ３１３に保存されるとともに、キー入力ＧＵＩ１８１と、データ更新Ｉ／Ｆモデル９に供給される。この結果、仮想モデル７１内のＶＫＤレジスタＶ４１０（図１５）の内容もＨＫＤ出力レジスタＨ３１３の内容と等価になる。本処理終了後は、ステップＳ４２４に移る。

【０２９６】ステップＳ８１４（ポート入力処理）で、入力ポートモデルＨ３０９（図１４）は、キー入力ＧＵＩ１８１からキーマトリクスの出力に相当する信号を受け取り、ＨＩＰレジスタＨ３１０（図１４）に保存するとともに、データ更新Ｉ／Ｆモデル９に供給する。この結果、仮想モデル７１内のＶＩＰレジスタＶ４０８（図１５）の内容もＨＩＰ入力レジスタＨ３１０の内容と等価になる。

【０２９７】また、ＨＩＰレジスタＨ３１０の内容は、ＣＰＵモデル４の読み出し命令によって、システムＩ／Ｆモデル８を介してＣＰＵモデル４に返される。この処理も、上述と同様、接続情報に基づいて行われる。本処理終了後は、ステップＳ４２４に移る。

【０２９８】ステップＳ４２４（仮想モデル書替？）で、データ更新Ｉ／Ｆモデル９は、機能関数選択テーブル３１（図１３、図１６）を参照して、ハードウェアシミュレーションＳ４０９の結果を仮想モデル７１に反映するか否かを判断する。本実施例では、機能関数ＩＯＷＲ（ＰＯＲＴＢ，Ｄ＊，Ｉ／Ｏ，ＷＲ）だけが仮想モデルシミュレーションＳ４０２を実行し、データ更新対象であるので、ステップＳ８１５の処理を実行した場合だけ「Ｙ」に進み、ステップＳ４２５の処理に移る。ステップＳ８１１，Ｓ８１４の処理を実行した場合は「Ｎ」に進み、ハードウェアモデルシミュレーションを終了する。

【０２９９】ステップＳ４２５（仮想モデルデータ更新起動）で、ハードウェアモデル２１は、更新すべきデータをデータ更新Ｉ／Ｆモデル９に供給し、これを仮想モデル７１（図１３）に渡す。仮想モデル７１は、データ更新Ｓ４２１（図１７）を起動して、所定のテーブルやレジスタの内容を更新する。

【０３００】以上の処理が終了すると、図１７のステップＳ４０９，Ｓ４２４，Ｓ４２５の処理を終了し、ステップ４１１の処理に移る。

【０３０１】図２０は、ハードウェアモデルデータ更新Ｓ４２３（図17）の詳細フローチャートを示す。

【０３０２】仮想モデルシミュレーションＳ４０２（図１７）を実行した結果、ハードウェアモデルデータ更新起動Ｓ４２０（図１８）が処理されると、仮想モデル７１は、更新すべきデータと、更新先の情報をデータ更新Ｉ／Ｆモデル９に供給し、図２０の開始に移る。

【０３０３】ステップＳ５０１（仮想→ハード・コード変換）で、データ更新Ｉ／Ｆモデル９は、受領した更新データをハードウェアモデルシミュレータ２－１（図１３）内のコード体系に変換する。例えば、仮想データメモリモデルＶ４０３内のＶＤＭテーブルＶ４０４（図１５）が書き替えられた場合、シンボルＤＭＥｍ，と更新データＤＡＴＡを「Ｃ＿ＨＤＭＥ、ＤＡＴＡ」に変換する。また、ＨＫＤ出力レジスタＨ３１３（図１４）が書き替えられた場合、デコード出力信号「Ｄ３，Ｄ２，Ｄ１，Ｄ０」を「Ｃ＿ＨＫＤＣ、Ｄ３，Ｄ２，Ｄ１，Ｄ０」に変換する。

【０３０４】仮想モデルシミュレータ７１とハードウェアモデルシミュレータ２－１とでコード体系が同じ場合には、本ステップを省略してもよい。処理終了後、ステップＳ５０３に移る。

【０３０５】ステップＳ５０３（Ｃ＿ＨＤＭＥ？）で、データ更新Ｉ／Ｆモデル９（図１３）は、受領したコードが「Ｃ＿ＨＤＭＥ」であるか否かを判断する。「Ｃ＿ＨＤＭＥ」であれば「Ｙ」に進み、ステップＳ５０９の処理に移る。「Ｃ＿ＨＤＭＥ」でなければ「Ｎ」に進み、ステップＳ５０７の処理に移る。

【０３０６】ステップＳ５０７（Ｃ＿ＨＫＤＣ？）で、データ更新Ｉ／Ｆモデル９は、受領したコードが「Ｃ＿ＨＫＤＣ」であるか否かを判断する。「Ｃ＿ＨＫＤＣ」であれば「Ｙ」に進み、ステップＳ５１３の処理に移る。「Ｃ＿ＨＫＤＣ」でなければ「Ｎ」に進み、ステップＳ５１４の処理に移る。

【０３０７】ステップＳ５１４（エラー処理）で、データ更新Ｉ／Ｆモデル９は、受領したコードがいずれにも該当しない場合には、表示装置（図２）にエラー発生の表示を行い、処理を終了する。

【０３０８】ステップＳ５０９で、データ更新Ｉ／Ｆモデル９は、コード「Ｃ＿ＨＤＭＥ、ＤＡＴＡ」を受領すると、データメモリモデルＨ３０３内のＨＤＭテーブルＨ３０４（図１４）を書き替え、処理を終了する。

【０３０９】ステップＳ５１３で、データ更新Ｉ／Ｆモデル９は、コード「Ｃ＿ＨＫＤＣ、Ｄ３，Ｄ２，Ｄ１，Ｄ０」を受領すると、出力ポート・デコーダモデルＨ３１１、Ｈ３１２内のＨＫＤテーブルＨ３１３（図１４）を書き替え、処理を終了する。

【０３１０】以上の処理が終了すると、図１７のステップＳ４２０を終了し、図９のステップＳ１０３またはＳ１０８に戻る。

【０３１１】このようにして、仮想モデルシミュレーションＳ４０２の結果がハードウェアモデル２１のテーブルやレジスタに反映される。なお、仮想モデル７１を使用するサブモデルで、データ更新が必要になるものは、図２０に示すデータ更新Ｓ４２３の処理に予め登録しておく必要がある。また、本実施例のように、仮想モデルを使わないデータ更新処理、例えばＳ５０９を余分に登録しておくこともできる。この結果、ハードウェアモデルデータ更新処理Ｓ４２３の内容を書き替えることなく、機能関数選択テーブル３１（図１３）の内容を書き

替えるだけで、データメモリモデルを仮想モデルシミュレーションに切り替えた場合にも即応できるようになる。

【０３１２】図２１は、仮想モデルデータ更新Ｓ４２１（図１７）の詳細フローチャートを示す。

【０３１３】ハードウェアモデルシミュレーションＳ４０９（図１７）を実行した結果、仮想モデルデータ更新起動Ｓ４２５（図１９）が処理されると、ハードウェアモデル２１（図１３）は、更新すべきデータと、更新先の情報をデータ更新Ｉ／Ｆモデル９に供給し、図２１の開始に移る。

【０３１４】ステップＳ７０１（ハード→仮想・コード変換）で、データ更新Ｉ／Ｆモデル９は、受領した更新データを仮想モデルシミュレータ７１（図１３）内のコード体系に変換する。例えば、データメモリモデルＨ３０３内のＨＤＭテーブルＨ３０４（図１４）が書き替えられた場合、アドレス信号ＡＤＲＳと更新データＤＡＴＡをシンボル「Ｃ＿ＤＭＥｍ」と「ＤＡＴＡ」に変換する。また、ＶＫＤ出力レジスタＶ４１０（図１５）が書き替えられた場合、デコード出力信号「Ｄ３，Ｄ２，Ｄ１，Ｄ０」を「Ｃ＿ＫＤＣ、Ｄ３，Ｄ２，Ｄ１，Ｄ０」に変換する。

【０３１５】仮想モデルシミュレータ７１とハードウェアモデルシミュレータ２－１とでコード体系が同じ場合には、本ステップを省略してもよい。処理終了後、ステップＳ７０３に移る。

【０３１６】ステップＳ７０３（Ｃ＿ＤＭＥ？）で、データ更新Ｉ／Ｆモデル９（図１３）は、受領したコードが「Ｃ＿ＤＭＥ」であるか否かを判断する。「Ｃ＿ＤＭＥ」であれば「Ｙ」に進み、ステップＳ７１０の処理に移る。「Ｃ＿ＤＭＥ」でなければ「Ｎ」に進み、ステップＳ７０７の処理に移る。

【０３１７】ステップＳ７０７（Ｃ＿ＫＤＣ？）で、データ更新Ｉ／Ｆモデル９は、受領したコードが「Ｃ＿ＫＤＣ」であるか否かを判断する。「Ｃ＿ＫＤＣ」であれば「Ｙ」に進み、ステップＳ７１４の処理に移る。「Ｃ＿ＫＤＣ」でなければ「Ｎ」に進み、ステップＳ７０８の処理に移る。

【０３１８】ステップＳ７０８（エラー処理）で、データ更新Ｉ／Ｆモデル９は、受領したコードがいずれにも該当しない場合には、表示装置（図２）にエラー発生の表示を行い、処理を終了する。

【０３１９】ステップＳ７１０（ＶＤＭテーブル書き替え処理）で、データ更新Ｉ／Ｆモデル９は、コード「Ｃ＿ＤＭＥ、ＤＡＴＡ」を受領すると、仮想データメモリモデルＶ４０３内のＶＤＭテーブルＶ４０４（図１５）を書き替え、処理を終了する。

【０３２０】ステップＳ７１４で、データ更新Ｉ／Ｆモデル９は、コード「Ｃ＿ＫＤＣ、Ｄ３，Ｄ２，Ｄ１，Ｄ０」を受領すると、仮想キーデコーダモデルＶ４０９内のＶＫＤテーブルＶ４１０（図１５）を書き替え、処理を終了する。

【０３２１】以上の処理が終了すると、図１７のステップＳ４２５を終了し、ステップＳ４１１に移る。このようにして、ハードウェアモデルシミュレーションＳ４０９の結果が仮想モデル７１のテーブルやレジスタに反映される。なお、ハードウェアモデル２１を使用するサブモデルで、データ更新が必要になるものは、図２１に示すデータ更新Ｓ４２１の処理に予め登録しておく必要がある。また、本実施例のように、仮想モデルを使わないデータ更新処理、例えばＳ７１０を余分に登録しておくこともできる。この結果、仮想モデルデータ更新処理Ｓ４２１の内容を書き替えることなく、機能関数選択テーブル３１（図１３）の内容を書き替えるだけで、仮想データメモリモデルをハードウェアモデルシミュレーションに切り替えた場合にも即応できるようになる。

【０３２２】［ターゲットプログラム例による動作の説明］図５および図８に示したプログラム例により、図１７～図２１のフローチャートを使用してその動作について説明する。第２実施例の特徴部分のみの説明に止めて第１実施例との重複を避けるため、図８のターゲットプログラム・リストの第１行と第２行について説明することとする。

【０３２３】第２実施例においても、図９に示したＣＰＵモデル４による処理フローはそのまま適用できる。

【０３２４】さて、先のプログラム例の第１行は、Ｉ／ＯポートＢモデル１４がデコーダモデル１５に、（ＰＢ１，ＰＢ０）＝（０，１）を出力し、デコーダモデル１５はキーマトリクスモデル１３に（Ｄ３，Ｄ２，Ｄ１，Ｄ０）＝（０，０，１，０）を出力するというものであったが、第２実施例では、これを仮想モデルシミュレーションで行い、その結果をハードウェアモデル２１に反映させる。また、先のプログラムの第２行は、キーマトリクスモデル１３の状態（ＰＡ３，ＰＡ２，ＰＡ１，ＰＡ０）＝（０，１，０，０）をＩ／ＯポートＡモデル１２に取り込むというものであったが、これをハードウェアシミュレーションにより行う。以下、各行ごとに詳細に説明する。

【０３２５】《ＲＯＭアドレス１００１Ｈの処理》図９のステップＳ１０１で、ＣＰＵモデル４は、プログラムカウンタＰＣの値「１００１Ｈ」を取得する。

【０３２６】ステップＳ１０２で、ＣＰＵモデル４は、機能関数ＦＥＴＣＨ（１００１Ｈ）を生成し、機能関数Ｉ／Ｆモデル５に渡す。

【０３２７】ステップＳ１０３で、周辺回路処理サブルーチンが呼び出される。

【０３２８】図１７のステップＳ４０１で、機能関数Ｉ／Ｆモデル５（図１３）は、機能関数選択テーブル３１（図１６）に関数が登録されていて、且つその引数の条件を満たしているか、チェックを行う（図１７）。ＦＥ

ＴＣＨ関数は登録されていて、アドレスの条件を満たしているので「Ｙ」に進み、機能関数を仮想モデル７１に渡す。

【０３２９】ステップＳ４０２で、仮想モデルシミュレーションＳ４０２を実行する。詳細フローチャートを図１８に示す。

【０３３０】図１８のＳ６０１で、仮想モデル７１は、機能関数Ｉ／Ｆモデル５から受領した機能関数ＦＥＴＣＨ（１００１Ｈ）を、仮想関数変換テーブル（図７相当）により、仮想関数「ＭＥＭ１００１」に変換する。

【０３３１】ステップＳ６０２で、仮想モデル７１は、仮想関数「ＭＥＭ１００１」がメモリを読み出すための関数（ＭＥＭ関数）か判断する。今の場合、ＭＥＭ関数なので「Ｙ」へ進む。

【０３３２】Ｓ６０３で、仮想モデル７１は、仮想プログラムメモリモデルＶ４０１内のＶＰＭテーブルＶ４０２（図１５）を参照し、ラベル「ＭＥＭ１００１」に対応する戻り値「Ａ２０１」（図５）を取得し、機能関数Ｉ／Ｆモデル５にこれを返す。なお、この場合は、仮想モデル７に書き込むわけではないため、ハードウェアデータ更新起動（Ｓ４２０）はバイパスされる。

【０３３３】以上で、仮想モデルシミュレーションＳ４０２が終了し、図９に戻る。

【０３３４】図９のステップＳ１０４で、機能関数Ｉ／Ｆモデル５はＣＰＵモデル４に戻り値「Ａ２０１」を渡す。ＣＰＵモデル４は、戻り値「Ａ２０１」を読み込み、命令レジスタ（不図示）に格納する。

【０３３５】ステップＳ１０５で、ＣＰＵモデル４は、戻り値「Ａ２０１」を解析し、「ＭＯＶ　ＰＯＲＴＢ，＃０１Ｈ」であると解読する。

【０３３６】ステップＳ１０６で、ＣＰＵモデル４は、解読した命令が周辺回路の読み書きに関するものかを判断し、「Ｙ」に進む。

【０３３７】ステップＳ１０７で、ＣＰＵモデル４は、解読した命令に対応する機能関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ、Ｉ／Ｏ、ＷＲ）を生成し、機能関数Ｉ／Ｆモデル５に渡す。

【０３３８】ステップＳ１０８で、周辺回路処理サブルーチン呼出す。

【０３３９】図１７のステップＳ４０１で、機能関数Ｉ／Ｆモデル５は、仮想モデル７１を呼び出すかどうかの判断を行う。ＩＯＷＲ関数は、図１６の機能関数選択テーブル３１に登録されていて、且つその引数の条件を満たしているので「Ｙ」に進むとともに、機能関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ）を仮想モデル７１に渡す。

【０３４０】ステップＳ４０２で、仮想モデルシミュレーションＳ４０２を実行する。

【０３４１】図１８のＳ６０１で、仮想モデル７１５は、仮想関数変換テーブル（図７相当）により、機能関数ＩＯＷＲ（ＰＯＲＴＢ，＃０１Ｈ）を仮想関数ＫＤＣ０１に変換する。

【０３４２】ステップＳ６０２で、「Ｎ」へ進む。

【０３４３】ステップＳ６０４で、「Ｎ」へ進む。

【０３４４】ステップＳ６１０で、「Ｎ」へ進む。

【０３４５】ステップＳ６１２で、「Ｙ」へ進む。

【０３４６】ステップＳ６１３で、仮想モデル７１の仮想キーデコーダモデルＶ４０９（図１５）でポート出力およびデコーダ処理を論理演算により行い、その結果をＶＫＤ出力レジスタＶ４１０に保存する。具体的には、第１実施例と同様に、デコーダ出力（Ｄ３，Ｄ２，Ｄ１，Ｄ０）＝（０，０，１，０）となる。これをキー入力ＧＵＩ１８１に出力し、キーマトリクスモデル１３の右から２本目の出力を「１」にする。

【０３４７】ステップＳ４２０で、この場合は、仮想モデル７１への書き込みを行ったので、ハードウェアモデルデータ更新Ｓ４２３を起動する。

【０３４８】図２０のステップＳ５０１で、データ更新Ｉ／Ｆモデル９は、ＶＫＤ出力レジスタＶ４１０から読み出した仮想コードＫＤＣ（００１０）をハードウェアコード「Ｃ＿ＨＫＤＣ（００１０）」に変換する。

【０３４９】ステップＳ５０３で、「Ｎ」へ進む。

【０３５０】ステップＳ５０７で、「Ｙ」へ進む。

【０３５１】ステップＳ５１３で、データ更新Ｉ／Ｆモデル９は、ハードウェアモデル２１のＨＫＤ出力レジスタＨ３１３（図１４）のデータを更新する。この結果、ハードウェアモデル２１のデコーダモデル１５の出力（Ｄ３，Ｄ２，Ｄ１，Ｄ０）＝（０，０，１，０）となり、仮想モデル７１側と整合することとなる。

【０３５２】以上で、図２０の処理を終了し、更に図１８の処理を終了し、図９に戻る。

【０３５３】図９のステップＳ１０９で、ＣＰＵモデル４は、機能関数Ｉ／Ｆモデル５からシミュレーション完了信号「ＤＯＮＥ」を受領する。

【０３５４】ステップＳ１１０で、ＣＰＵモデルシミュレータ４は、「Ｎ」へ進み、ステップＳ１０１へ戻る。

【０３５５】以上で、プログラムの第１行目のシミュレーションが終了する。

【０３５６】《ＲＯＭアドレス１００２Ｈの処理》図９のステップＳ１０１～Ｓ１０６で、ＰＣの値「１００２Ｈ」取得から、「ＭＯＶ　Ａ，ＰＯＲＴＡ」との命令解読までは、第１行目と同様の手順のため説明は省略する。

【０３５７】ステップＳ１０７で、ＣＰＵモデル４は、解読した命令に対応する機能関数ＩＯＲＤ（ＰＯＲＴＡ，Ｉ／Ｏ，ＲＤ）を生成し、機能関数Ｉ／Ｆモデル５に渡す。

【０３５８】ステップＳ１０８で、周辺回路処理サブルーチンを呼出す。

【０３５９】図１７のステップＳ４０１で、この機能関数ＩＯＲＤ（ＰＯＲＴＡ，Ｉ／Ｏ，ＲＤ）は図１６の機

能関数選択テーブル３１に登録されていないので「Ｎ」へ進む。

【０３６０】ステップＳ４０３（バスＩ／Ｆモデルシミュレーション）で、バスＩ／Ｆモデル６（図１３）は、機能関数ＩＯＲＤ（ＰＯＲＴＡ，Ｉ／Ｏ，ＲＤ）を端子信号「ＡＤＲＳ，／ＣＳ，／ＲＤ」などに変換する。

【０３６１】ステップＳ４１０（シミュレータ間Ｉ／Ｆ処理）で、バスＩ／Ｆモデル６（図１３）は、端子信号名をハードウェアシミュレータ２－１対応に変換したり、データ授受のタイミングを調整する。

【０３６２】ステップＳ４１１（システムＩ／Ｆモデルシミュレーション）で、システムＩ／Ｆモデル８は、端子信号（図１２）を受け取る。端子信号の授受動作は第１実施例と同様であるからその説明は省略する。

【０３６３】ステップＳ４０９で、ハードウェアシミュレーションを実行する。

【０３６４】以下、ハードウェアシミュレーションＳ４０９の詳細を説明する。

【０３６５】図１９のステップＳ８０１で、ハードウェアモデル２１（図１３）は、端子信号ＭＥＭが「０」か、すなわちメモリの読み書きサイクルかを判断し、「Ｎ」へ進む。

【０３６６】ステップＳ８０４で、ハードウェアモデル２１は、端子信号Ｉ／Ｏが「０」か、すなわちメモリ以外の周辺回路の読み書きサイクルか判断し、「Ｙ」へ進む。

【０３６７】ステップＳ８０７で、アドレス信号ＡＤＲＳが「ＰＯＲＴＡ」か判断し、「Ｙ」へ進む。

【０３６８】ステップＳ８１４（ポート入力処理）で、入力ポートモデルＨ３０９は、接続情報に基づき各接続点のレベルを演算によって求める。いま、キー入力ＧＵＩ１８１のキーマトリクスの右から２番目で上から３番目のスイッチが押下されているとすると、デコーダ出力Ｄ１は「１」であるので、入力ポートＡモデル１２の入力ＰＡ２も「１」となる。従って、入力ポートＡに（ＰＡ３，ＰＡ２，ＰＡ１，ＰＡ０）＝（０，１，０，０）を得る。入力ポートモデルＨ３０９（図１４）のデータは、ＨＩＰ入力レジスタ３１０に記憶される。

【０３６９】ステップＳ４２４（仮想モデル書替？）で、ハードウェアシミュレーションＳ４０９の結果を仮想モデルに反映するか判断する。ＰＯＲＴＡ（　）は、図１６の機能関数選択テーブル３１にないので、「Ｎ」へ進む。

【０３７０】以上で、図１９の処理を終了し、図１７の処理に戻る。

【０３７１】図１７のステップＳ４１１（システムＩ／Ｆモデルシミュレーション）で、システムＩ／Ｆモデル８は、ＨＩＰ入力レジスタ３１０からシミュレーション結果を読み出して、端子信号出力（０，１，０，０）即ち「＃０４Ｈ」を、ステップＳ４１０（シミュレータ間Ｉ／Ｆ処理）を介してバスＩ／Ｆモデル６に渡す。

【０３７２】ステップＳ４０３で、バスＩ／Ｆモデル６は、「＃０４Ｈ」を受領し、機能関数Ｉ／Ｆモデル５に渡し、更にＣＰＵモデル４に渡す。

【０３７３】図９のステップＳ１０９で、ＣＰＵモデル４は、機能関数Ｉ／Ｆモデル５より受領した「＃０４Ｈ」をレジスタＡに格納する。

【０３７４】このように、第２実施例によれば、第１実施例と同様、仮想モデルを使用することで、ハードウェアモデルシミュレーションに要する時間を大幅に短縮できるとともに、１つの周辺回路のシミュレーション結果が、他の周辺回路のシミュレーション結果に影響を及ぼす場合であっても、仮想モデルのシミュレーション結果をハードウェアモデルに反映したり、ハードウェアモデルのシミュレーション結果を仮想モデルに反映させることで、シミュレーション結果に誤検証を生ずることなく、正確なシミュレーションができるようになる。

【０３７５】また、第１実施例のように、仮想モデルを使用できる条件として、１つの周辺回路のシミュレーション結果が、他の周辺回路のシミュレーション結果に影響を及ぼさないという制約がなくなるので、仮想モデルとして登録できるサブモデルの種類を第１実施例に比べて増やすことができる。

【０３７６】また、ハードウェアシミュレータを使用する対象を、仮想関数選択テーブルの内容を書き替えるだけで、簡単に選択することができる。この結果、問題の残ったハードウェアモデルだけを集中して検証することができるようになり、短期間でターゲットプログラムや周辺回路の問題を解決することができる。例えば、キー入力モデルのいずれかに異常があるが、出力ポート側は正常に動作していることが確かめられれば、出力ポート側を仮想モデルに登録することで、デコード出力までのシミュレーション時間を短縮し、入力ポート側を重点的に検証することができる。

【０３７７】［第３実施例］第１実施例および第２実施例では、ハードウェアシミュレーションはマイコンシミュレータからの周辺回路処理ルーチンとして位置付けされ、ＩＯＲＤ，ＩＯＷＲ等入出力に関する機能関数により呼出されていた。しかしながら、例えば通信回線からのデータ受信等外部要因による事象を受け入れる電子機器もあり得る。第３実施例は、このような電子機器に対して有効なシステムシミュレーションを実行できるシステムシミュレータおよびシステムシミュレーション方法を提供するものである。すなわち、ハードウェアモデルの外部端子を監視し、状況変化をキャッチするとハードウェアシミュレーションを起動する。このときのハードウェアシミュレーションは周辺回路処理サブルーチンにおけるハードウェアシミュレーションに対する割り込みとして優先的に扱われる。ハードウェアシミュレーションの結果は、必要に応じて仮想モデルの更新に反映さ

れ、以後仮想モデルシミュレーションによって代替できるような構成とすることにより、システムシミュレーションの高速化を図るのである。

【０３７８】図２２は、本発明のシステムシミュレータの第３実施例の機能ブロック図である。図２２と図１３を対比すれば、第３実施例は第２実施例におけるハードウェアシミュレータ２－１に割込Ｉ／Ｆモデル２４とバス要求Ｉ／Ｆモデル２５とを付加した点が異なっているのがわかる。また、シミュレーション装置のＳＩＯ端子２３およびシリアル入出力Ｉ／ＦモデルＳ＿ＳＩＯ（図２）を顕示した。割込Ｉ／Ｆモデル２４とバス要求Ｉ／Ｆモデル２５はＣＰＵモデル４２と接続されてシミュレータ間Ｉ／Ｆモデルを構成するが、それらの接続端子信号はターゲットプログラムに基づく信号変化とは無関係に変化するため、バスＩ／Ｆモデル６およびシステムＩ／Ｆモデル８とは別個に設けたものである。

【０３７９】割込Ｉ／Ｆモデル２４は、ハードウェアモデル２２からの割込み要求信号をＣＰＵモデル４２に伝えたり、ＣＰＵモデル４２からハードウェアモデル２２側に割込み許可信号を伝えたりする。

【０３８０】バス要求Ｉ／Ｆモデル２５は、ハードウェアモデル２２からのバス開放要求信号をＣＰＵモデル４２に伝えたり、ＣＰＵモデル４２からハードウェアモデル２２側にバス使用許可信号を伝える。

【０３８１】シリアル入出力Ｉ／ＦモデルＳ＿ＳＩＯは、ＳＩＯ端子２３から入力される信号を復調して仮想モデル７２あるいは所定の記憶領域Ｓ＿ＲＡＭ（図２）に渡し、また仮想モデル７２あるいは所定の記憶領域Ｓ＿ＲＡＭから入力される信号を変調してＳＩＯ端子２３に出力する。

【０３８２】第３実施例においても、図２に示したシミュレーション装置のブロック図およびその説明は第１実施例および第２実施例と同様に適用できる。さらに、これまでの実施例では使用されることがなかった割込制御装置Ｓ＿ＩＮＴ，ＤＭＡ装置Ｓ＿ＤＭＡおよびシリアル入出力Ｉ／ＦモデルＳ＿ＳＩＯが有用になる。

【０３８３】図２３は第３実施例におけるハードウェアモデル２２の構成図であり、図１４に示した第２実施例におけるハードウェアモデル２１に対して、ＨＤＡ設定レジスタＨ３０６付きＤＭＡモデルＨ３０５、およびＨＳＩ設定レジスタＨ３０８付きＳＩＯモデルＨ３０７が付加されている。ＤＭＡモデルＨ３０５，ＳＩＯモデルＨ３０７は、システムＩ／Ｆモデル８と接続されるとともに、それぞれバス要求Ｉ／Ｆモデル２５，割込Ｉ／Ｆモデル２４と接続される。また、ＨＤＡ設定レジスタＨ３０６，ＨＳＩ設定レジスタＨ３０８は、データ更新Ｉ／Ｆモデル９と接続され、それぞれＤＭＡモデルＨ３０５，ＳＩＯモデルＨ３０７の更新のために使用されるとともに、仮想モデル７２の更新のために出力される。

【０３８４】ＳＩＯモデルＨ３０７は、ＳＩＯ端子２３と接続され、ＳＩＯ端子２３は図示しない通信端末と接続され、通信データを授受する。ＳＩＯモデルＨ３０７は、通信端末から１バイト分のデータを受信すると、割込み要求信号を割込みＩ／Ｆモデル２４を介してＣＰＵモデル４２に出力する。さらに、受信データをデータメモリモデルＨ３０３にＤＭＡ転送するため、ＤＭＡモデルＨ３０５にデータ転送要求信号ＤＲＱを出力する。ＤＭＡモデルＨ３０５にデータ転送要求信号ＤＲＱが受け付けられると、ＳＩＯモデルＨ３０７は、１バイトの受信データをＤＭＡモデルＨ３０５に出力する。これらの処理はいずれも、入力端子に所定の端子信号を与え、接続情報に基づき各接続点および出力端子の信号レベルを演算により求めることにより行われる。

【０３８５】ＨＳＩ設定レジスタＨ３０８には、ＣＰＵモデル４２のストア命令により通信速度やパリティなどのパラメータを設定することができる。

【０３８６】ＤＭＡモデルＨ３０５は、ＣＰＵモデル４２を介さずにデータ転送を可能にするもので、データメモリモデルＨ３０３の所定の領域の記憶内容を他の領域に転送したり、ＳＩＯモデルＨ３０７との間でデータ転送を行う。本実施例では、ＳＩＯモデルＨ３０７で受信したデータをデータメモリモデルＨ３０３に転送する場合を例に説明する。

【０３８７】ＤＭＡモデルＨ３０５は、ＳＩＯモデルＨ３０７からデータ転送要求信号ＤＲＱを受け付けると、ＣＰＵモデル４２にバス開放要求信号をバス要求Ｉ／Ｆモデル２５を介してＣＰＵモデル４２に出力する。ＣＰＵモデル４２は、バス開放要求信号ＨＲＱを受け付けると、バス使用許可信号ＨＬＤＡをバス要求Ｉ／Ｆモデル２５を介してＤＭＡモデルＨ３０５に返すとともに、バスを開放する。ＤＭＡモデルＨ３０５は、バス使用許可信号ＨＬＤＡを確認すると、ＳＩＯモデルＨ３０７にデータ転送許可信号ＤＡＣＫを返す。ＳＩＯモデルＨ３０７は、データ転送許可信号ＤＡＣＫを確認すると、１バイトの受信データをＤＭＡモデルＨ３０５に出力する。ＤＭＡモデルＨ３０５は受信データを一旦ＤＨＡ設定レジスタＨ３０６に格納した後、データメモリモデルＨ３０３の所定のアドレスに受信データを書き込む。以上の処理を繰り返し、ＤＭＡ転送が終了すると、ＤＭＡモデルＨ３０５は、バス開放要求信号を非活性化してバス要求Ｉ／Ｆモデル２５を介してＣＰＵモデル４２に出力する。ＣＰＵモデル４２は、非活性化したバス開放要求信号ＨＲＱを確認すると、非活性化したバス使用許可信号ＨＬＤＡをバス要求Ｉ／Ｆモデル２５を介してＤＭＡモデルＨ３０５に返すとともに、バス使用権を復活する。以上の処理は接続方法と端子信号をもとに演算されて、所定の結果を得る。

【０３８８】図２４は、第３実施例における仮想モデル７２の構成図であり、図１５に示した第２実施例における仮想モデル７１に対して、ＶＲＴテーブルＶ４０６付

き仮想通信モデルＶ４０５が付加されている点が異なる。

【０３８９】仮想通信モデルＶ４０５は、図２３におけるＤＭＡモデルＨ３０５およびＳＩＯモデルＨ３０７に対応し、またＶＲＴテーブルＶ４０６は図２３におけるＨＤＡ設定レジスタＨ３０６およびＨＳＩ設定レジスタＨ３０８に対応する。

【０３９０】仮想通信モデルＶ４０５は、ＳＩＯ端子２３を介して受信した信号を復調して、データメモリモデルのＨＤＭテーブルＨ３０４またはＶＤＭテーブルＶ４０４に記憶させたり、データメモリモデルのＨＤＭテーブルＨ３０４またはＶＤＭテーブルＶ４０４の記憶内容を変調してＳＩＯ端子２３を介して送信できればよい。例えば、通信相手側の変調する前の文字データを直接仮想データメモリＶ４０３に書き込むようにしてもよいし、あるいは、別に構成したモデム（ＭＯＤＥＭ）で受信・復調した文字データを図２に示すＳ＿ＤＭＡを利用して仮想データメモリＶ４０３にＤＭＡ転送するようにしてもよい。本実施例では、シミュレーション装置が有するＳ＿ＳＩＯとＳ＿ＤＭＡを使用して、所定の記憶領域ＨＤＭテーブルＨ３０４またはＶＤＭテーブルＶ４０４に転送する例を説明する。

【０３９１】ＶＲＴテーブルＶ４０６には、ＳＩＯモデルの通信速度や通信モード、パリティなどの通信パラメータと、ＤＭＡの転送元アドレスや転送先アドレスと、割込み制御装置Ｓ＿ＩＮＴの割込みベクタなどを設定する。

【０３９２】シリアル入出力Ｉ／Ｆ装置Ｓ＿ＳＩＯは、ＳＩＯ端子２３と接続され、通信相手側と通信データを授受する。また、シリアル入出力Ｉ／Ｆ装置Ｓ＿ＳＩＯは、ＣＰＵモデル４２のストア命令により仮想通信モデルＶ４０５を介して、通信速度や通信モード、パリティなどが設定される。シリアル入出力Ｉ／Ｆ装置Ｓ＿ＳＩＯが１バイト分の通信データを受信すると、シリアル入出力Ｉ／Ｆ装置Ｓ＿ＳＩＯは割込み制御装置Ｓ＿ＩＮＴ（図２）に割込み要求信号を出力する。割込み処理が起動されると、ＤＭＡ装置Ｓ＿ＤＭＡは、シリアル入出力Ｉ／Ｆ装置Ｓ＿ＳＩＯから通信データを読み込んで、メモリＳ＿ＲＡＭ内に形成された所定の記憶領域ＶＤＭテーブルＶ４０４にＤＭＡ転送を開始する。

【０３９３】ＤＭＡ転送が終了すると、ＶＤＭテーブルＶ４０４の内容は、ハードウェアモデル２２に渡され、データ更新処理によりＨＤＭテーブルＨ３０４の内容が更新される。

【０３９４】第２実施例では、図１４と図１５とを見比べれば明らかなように、ハードウェアモデルを構成する各モデルと、仮想モデルを構成する各モデルとは１：１対応の関係にあるが、第１，第３実施例では、上述のように２：１対応の関係にある。即ち、ハードウェアモデルを構成する複数のサブモデルをまとめた機能を１つの仮想モデルで実現することができる。

【０３９５】図２５は、機能関数選択テーブル３２の内容例であり、第１実施例および第２実施例における機能関数選択テーブル３，３１の内容（図６，図１６）と対比すると、フラグが設けられている点に特徴がある。シミュレータ操作者が、図示しない編集装置を使って、このフラグを書き替えることができる。フラグが「１」であって引数の条件に合えば、仮想モデルを選択することができ、「０」であれば、仮想モデルを選択することができない。

【０３９６】このように、フラグを書き替えるだけで、仮想モデルを使うか否かを決定することができるので、第１や第２実施例のように機能関数選択テーブル全体を書き替える場合に比べ、ハードウェアモデルと仮想モデルの選択の設定が容易にできるようになる。

【０３９７】図２５（Ａ）は、ハードウェアモデル２２でＳＩＯ，ＤＭＡ転送を行う場合、図２５（Ｂ）は、仮想モデル７２で受信したシリアルデータを受信し、ハードウェアモデル２２のデータを更新する場合の機能関数選択テーブル３２の内容例を示す。

【０３９８】ここで、第３実施例における、ＣＰＵモデルシミュレーション，周辺回路処理サブルーチンおよびハードウェアモデルシミュレーションの特徴部分について、図２７，図２８および図２９により説明し、以下の動作説明の流れを円滑にする。

【０３９９】先ず、図２７のＣＰＵモデルシミュレーションのフローチャートを参照すると、本フローチャートは、図９のフローチャートに、割込み要求の有無をチェックするステップ（Ｓ３６１），バス開放要求の有無をチェックするステップ（Ｓ３６２），割込みに関するステップ（Ｓ３６７，Ｓ３６８）およびバス開放に関するステップ（Ｓ３６３～Ｓ３６６）が付加されている。

【０４００】割込み要求の有無やバス開放要求の有無は、ＣＰＵモデル４２が割込み端子，バス要求端子に割込み要求信号や，バス開放要求信号が入力されているか否かを確認することにより検知する。本例では、説明を簡単にするために、スタート直後に、このような確認するステップＳ３６１，Ｓ３６２を設けたが、実際には、ＣＰＵモデルシミュレーションの処理とは独立して、割込み的に要求を受け付けるようにしている。

【０４０１】ステップＳ３６１（割込み要求有？）で、ハードウェアモデル２２のいずれかのサブモデルが割込み要求信号を発生し、割込みＩ／Ｆモデル２４を介してＣＰＵモデル４２に入力されたか否かを判断する。割込み要求信号が有効であれば「Ｙ」に進み、ステップ３６７の処理に移る。割込み要求信号が有効でなければ「Ｎ」に進み、ステップ３６２の処理に移る。

【０４０２】ステップＳ３６７（割込み受付可？）で、ＣＰＵモデル４２は、割込み要求が受付可能かどうかチェックし、受付可能であれば、「Ｙ」へ進み、ステップ

Ｓ３６８で所定の割込み処理を行う。本例では、説明を簡単にするために、ＳＩＯモデルＨ３０７だけから割込みがかかり、ＤＭＡモデルＨ３０５を起動する例を説明するが、実際には、割込み要求は多くの周辺回路から発せられるので、割込みコントローラ（図２の割込制御装置Ｓ＿ＩＮＴに相当）を設けて、割込み要求の調停をすることもある。割込みコントローラは、ＣＰＵモデル４２に対して割込み要求と、優先順位の高い方の割込みベクタ（どこから割込み要求があって、どの割込み処理をするかを指示するもの）を出力する。ＣＰＵモデル４２が割込みベクタに対応する割込み処理を実行する。割込み処理は、割込みベクタに対応する割込み処理が記述されたアドレスをプログラムカウンタＰＣにセットして、図２７の流れ図を呼び出すことで、再帰的に処理される。

【０４０３】ステップＳ３６２（バス開放要求有？）で、バス開放要求信号（ＨＲＱ：ホールドリクエスト）が入力された場合、ＣＰＵモデル４２はそれが受付可能かどうかをチェックし、受付可能であれば、「Ｙ」に進み、ステップＳ３６３の処理に移る。受付不可能であれば、「Ｎ」に進み、ステップＳ１０１の処理に移る。

【０４０４】ステップＳ３６４（バス使用許可）で、ＣＰＵモデル４２は、以下のようにバスを開放する処理を行う。本例では、説明を簡単にするために、ＤＭＡモデルＨ３０５だけからバス要求がかかり、ＤＭＡモデルＨ３０５がバスを占有する例を説明するが、実際には、バス要求は多くの周辺回路から発せられるので、バスアービタ（不図示）を設けて、バス要求の調停をすることもある。

【０４０５】ＣＰＵモデル４２は、バス要求を受け付けると、バス使用許可信号（ＨＬＤＡ：ホールド　ＡＣＫ）をバス要求Ｉ／Ｆモデルを介してＤＭＡモデルＨ３０５に返す。

【０４０６】ステップＳ３６５（バス開放）で、ＣＰＵモデル４２は、バスを開放して、バス使用権をＤＭＡモデルＨ３０５に渡す。通常、ＣＰＵモデル４２がバスを開放すると、ＣＰＵモデルは何も処理できなくなる。

【０４０７】ステップＳ３６６（バス開放？）で、ＣＰＵモデル４２は、バス開放要求信号が非活性化するのを待つ。即ち、ＤＭＡモデルＨ３０５がＤＭＡ処理中で、バス開放要求信号が活性化している間は、「Ｎ」に進み、非活性化したことを検出すると、「Ｙ」に進み、ステップＳ１１０に移る。

【０４０８】ステップＳ１１０（ブレーク？）で、処理停止または中断の命令でなければ、ＣＰＵモデル４２は「Ｎ」へ進み、ステップＳ３６１～Ｓ１０９の処理を繰り返す。処理停止または中断の命令であれば、、ＣＰＵモデル４２は「Ｙ」へ進み、処理を中断または終了する。

【０４０９】図２８は、第３実施例における周辺回路処理サブルーチンのフローチャートを示し、図２９は、図２８のステップＳ４１３（ＣＰＵ特殊端子入出力）の詳細フローチャートを示すものである。

【０４１０】次に、図２８と図２９を参照すると、本フローチャートは、図１７のシステムＩ／Ｆモデルシミュレーションステップ４１１を詳細化するとともに、図１７のフローチャートに、「外部端子変化？」（Ｓ２１６），「割込み要求有り？」（Ｓ４３２），「割込み要求信号出力」（Ｓ４３３），「バス要求有？」（Ｓ４１７），「バスホールド出力」（Ｓ４１８）および「ホールドアクノリッジ」（Ｓ４１９）を追加したことがわかる。

【０４１１】ステップＳ４３１（仮想モデル呼出し？）で、マイコンシミュレータ１－２（図２２）は、機能関数選択テーブル３２を参照して、仮想モデルシミュレータ７２、またはハードウェアシミュレータ２－２のいずれを使用するかを判断する。第１、第２実施例では、機能関数選択テーブルに検証すべき機能関数が登録されているか、登録されていれば引数条件を満たしているかを判断して、いずれかのシミュレータ１－２，２－２を選択していた。本実施例においては、機能関数選択テーブル３２内にフラグを設け、フラグの有無で、シミュレータ１－２，２－２のいずれを選択するか判断するようにした。

【０４１２】具体的には、マイコンシミュレータ１－２は、検証すべき機能関数のフラグが「１」で、引数条件を満たしていれば、「Ｙ」へ進み、仮想モデルシミュレータ７２を使用し、ステップＳ４２２（仮想モデルシミュレーション）を実行する。逆に、フラグが「０」であれば無条件に「Ｎ」に進み、ハードウェアシミュレータ２－２は、ステップ４１２（ハードウェアモデルシミュレーション）を実行すべく処理を進める。即ち、ステップＳ４０３（バスＩ／Ｆモデルシミュレーション）で、バスＩ／Ｆモデル６（図２２）は、機能関数Ｉ／Ｆモデル５から機能関数を受け取り、端子信号に変換し、ステップＳ４１０（シミュレータ間Ｉ／Ｆモデルシミュレーション）でシミュレータ間コード変換を行い、システムＩ／Ｆモデル８に端子信号を供給する。ステップＳ４１１（システムＩ／Ｆモデルシミュレーション）で、システムＩ／Ｆモデル８は、端子信号を受け取り、これをハードウェアモデル２２に渡す。

【０４１３】ステップＳ２１２～Ｓ２１５およびステップＳ２１６がＩ／Ｆモデルシミュレーション４１１を構成する。ステップＳ２１２～Ｓ２１５は、第１実施例の図１０と同様に動作する。

【０４１４】ステップＳ２１６（外部端子変化？）で、ハードウェアモデル２２は、ハードウェアモデル２２に接続された外部端子（ＳＩＯ端子２３など）の状態を定期的に確認し、変化があれば、ハードウェアモデルシミュレーション４１２内の所定のサブモデルシミュレーションを起動する。例えば、ハードウェアモデル２２

（図２３）は、ＳＩＯモデルＨ３０７のＳＩＯ端子２３からの入力信号の状態を定期的に確認し、変化があれば、ＳＩＯモデルＨ３０７のシミュレーションを起動する。

【０４１５】なお、図２８において、端子信号変化（Ｓ２１３）を先に検出し、「システムＩ／Ｆ完了？」（Ｓ２１５）で全端子信号の受信が完了しない間は、ＣＰＵモデル４２からの信号受信を優先させ、「外部端子変化？」（Ｓ２１６）の処理をマスクするようにしてもよい。

【０４１６】このように、第２実施例では、ＣＰＵモデル４２から発せられる端子信号だけに応じてハードウェアシミュレータ２－１を起動するようにしていたのに対し、本例では、このように、外部端子が変化したかをチェックし、変化したらハードウェアシミュレーションを開始するようにした。

【０４１７】ステップＳ４１２（ハードウェアモデルシミュレーション）の結果、割込要求信号やバス要求信号などＣＰＵ特殊端子信号が活性化した場合には、ハードウェアシミュレータ２－２は、ステップＳ４１３（ＣＰＵ特殊端子入出力）を処理する。非活性のままであれば、ステップ４２４，Ｓ４２５で仮想モデルデータ更新の処理に移る。

【０４１８】ステップＳ４１３の具体的な処理の流れを図２９を参照して説明する。

【０４１９】ステップＳ４３２（割込み要求信号有り？）で、ハードウェアモデルシミュレーション（Ｓ４１２）の結果、ハードウェアモデル２２（図２２）は、割込要求信号ＩＲＱが活性化したかどうかを判断する。ＩＲＱが活性化していれば、「Ｙ」へ進み、ステップＳ４２３の処理に移り、ＩＲＱが非活性のままであれば、「Ｎ」へ進み、ステップＳ４１７の処理に移る。

【０４２０】ステップＳ４２３で、ハードウェアモデル２２内のＳＩＯモデルＨ３０７（図２３）は、割込みＩ／Ｆモデル２４経由でＣＰＵモデル４２に対して割込み要求信号ＩＲＱを出力する。

【０４２１】ステップＳ４１７（バス要求有り？）で、ハードウェアモデル２２（図２２）は、ハードウェアモデルシミュレーション（Ｓ４１２）の結果、バス要求信号ＨＲＱが活性化したかどうかを判断する。ＨＲＱが活性化していれば、「Ｙ」へ進み、ステップＳ４１８の処理に移り、ＨＲＱが非活性のままであれば、「Ｎ」へ進み、ステップＳ４２４の処理に移る。

【０４２２】ステップＳ４１８（バス開放要求出力）で、ハードウェアモデル２２内のＤＭＡモデルＨ３０５（図２３）は、バス要求Ｉ／Ｆモデル２５経由でＣＰＵモデル４２に対してバス開放要求信号ＨＲＱを出力する。

【０４２３】図２７のステップＳ３６２で、ＣＰＵモデル４２は、バス開放要求信号ＨＲＱが活性化したことを検出すると、Ｓ３６３→Ｓ３６４→Ｓ３６５と進み、バス使用権を放棄するとともに、バス使用許可信号ＨＬＤＡを活性化して、バス要求Ｉ／Ｆモデル２５経由でＤＭＡモデルＨ３０５に返す。

【０４２４】ステップＳ４１９で、ＤＭＡモデルＨ３０５（図２３）は、バス要求Ｉ／Ｆモデル２５経由でＣＰＵモデル４２からバス使用許可信号ＨＬＤＡを受領したか否かを判断する。ＨＬＤＡを受領しなければ、「Ｎ」へ進み、ステップＳ４１９に留まる。ＨＬＤＡを受領すれば、バス使用権を取得して「Ｙ」へ進み、ステップＳ４１２に戻り、ＤＭＡモデルＨ３０５のシミュレーションを開始する。

【０４２５】ステップＳ４１２で、ＤＭＡモデルＨ３０５のシミュレーションが終了すると、ＤＭＡモデルＨ３０５は、バス開放要求信号ＨＲＱを非活性化した信号を出力するとともに、バスを開放する。以後、Ｓ４３２→Ｓ４１７→Ｓ４１８と進む。

【０４２６】ステップＳ４１８（バス開放要求出力）で、ハードウェアモデル２２内のＤＭＡモデルＨ３０５（図２３）は、バス要求Ｉ／Ｆモデル２５経由でＣＰＵモデル４２に対して、非活性化したバス開放要求信号ＨＲＱを出力する。

【０４２７】図２７のステップＳ３６６で、ＣＰＵモデル４２は、バス開放要求信号ＨＲＱが非活性化したことを検出すると、バス使用権を取得するとともに、バス使用許可信号ＨＬＤＡを非活性化して、バス要求Ｉ／Ｆモデル２５経由でＤＭＡモデルＨ３０５に返す。

【０４２８】図２９のステップＳ４１９で、ＤＭＡモデルＨ３０５（図２３）は、バス要求Ｉ／Ｆモデル２５経由でＣＰＵモデル４２から、非活性化したバス使用許可信号ＨＬＤＡを受領したか否かを判断する。ＨＬＤＡを受領しなければ、「Ｎ」へ進み、ステップＳ４１９に留まる。非活性化したＨＬＤＡを受領すれば、「Ｙ」へ進み、ステップＳ４１２に戻る。

【０４２９】ステップＳ４３２，Ｓ４１７でいずれも「Ｎ」であったとき、ハードウェアモデル２２は、ステップＳ４２４に進む。

【０４３０】ステップＳ４２４（仮想モデルデータ更新要？）で、ハードウェアモデル２２は、ハードウェアモデルシミュレーション（Ｓ４１２）の結果が仮想モデル７２に関係しているか否かを機能関数選択テーブル３２のフラグを参照して判断する。ハードウェアモデル２２は、ハードウェアモデルシミュレーション（Ｓ４１２）に対応する機能関数が機能関数選択テーブル３２に登録されていて、かつその引数条件を満たし、フラグが「１」であれば、「Ｙ」へ進み、ステップＳ４２５の処理に移る。対応する機能関数のフラグが「０」であれば、引数条件に関係なく「Ｎ」へ進み、ステップＳ２１２の処理に移る。

【０４３１】ステップＳ４２５，Ｓ４２１の処理は第２

実施例と同じであり、説明を省略する。

【０４３２】ステップＳ４０３，Ｓ４２０，Ｓ４２１は、第２実施例の図１７と同じであり、また、ステップＳ２０５～Ｓ２０８は、第１実施例の図１０と同じであり、説明を省略する。

【０４３３】次に、第３実施例の動作を、図２６に示すターゲットプログラムを例にとり、図２７～図３３のフローチャートおよび図３４のタイミングチャートを用いて説明する。以下の説明は、重複を回避するため、第１実施例および第２実施例について行ったようなプログラム１行ごとに詳述することはせず、より要約的なものに止める。

【０４３４】本プログラムは、ＳＩＯ端子２３で受信したデータをＲＡＭにＤＭＡ転送するというものである。第１のケースとして、ハードウェアシミュレータ２－２によって行う場合と、第２のケースとして、仮想モデル７２を用いて行う場合とについて述べる。

【０４３５】［第１のケース：ハードウェアシミュレータ２－２］

《ターゲットプログラムＴ３０１：ＳＩＯ初期設定の処理》図２６において、先ずＣＰＵモデル４２は、ハードウェアモデル２２のＳＩＯモデルＨ３０７（図２３）に通信速度，パリティ有無，受信モード等を設定する。この設定は、先の実施例と同様に、アドレスとしてＳＩＯモデルＨ３０７を指定したＭＯＶ命令（図８）により行えばよい。

【０４３６】《ターゲットプログラムＴ３０２：ＤＭＡ初期設定の処理》ＣＰＵモデル４２は、同様にして、ＤＭＡモデルＨ３０５（図２３）に、使用するチャネル番号，転送モード，アドレス増／減の別や、ＤＭＡ転送元およびＤＭＡ転送先（データメモリモデル）の各開始アドレスと終了アドレス等を設定する。

【０４３７】《ターゲットプログラムＴ３０３：ＳＩＯ割込み許可》ＣＰＵモデル４２は、ＳＩＯモデルＨ３０７に割込みの許可を与える。ＳＩＯモデルＨ３０７（図２３）は、ＳＩＯ端子２３に入力されるスタートビットが変化したか否かを検出する処理を開始する。

【０４３８】以上の処理Ｔ３０１，Ｔ３０２およびＴ３０３は、図２７において、Ｓ３６１とＳ３６２では「Ｎ」であるから、ステップＳ１０１～Ｓ１１０を3回繰返すことにより実行される。そして、ステップＳ１２３およびＳ1-2８では、図２８の周辺回路処理サブルーチンが起動されるが、ステップＳ２１６（外部端子変化？），Ｓ４３２（割込み要求有り？）およびＳ４１７（バス要求有り？）では、いずれも「Ｎ」であるから、図１７と同一の処理ルートを辿ることになる。

【０４３９】《ターゲットプログラムＴ３０４：割込み発生》ＣＰＵモデル４２は、Ｔ３０３以降の図示しない処理を継続して実行する。

【０４４０】ここで、ＳＩＯ端子２３に調歩同期式により通信データが入力されたとする。

【０４４１】時刻ｔ０（図３０）のステップＳ２１６（図２８の外部端子変化？）で、ＳＩＯモデルＨ３０７は、ＳＩＯ端子２３が「１」から「０」に変化するとスタートビットであると判断する。ＳＩＯモデルＨ３０７は、このスタートビットを検出すると（図３０（ｂ））、「Ｙ」へ進み、ハードウェアモデルシミュレーション（Ｓ４１２）を起動する。ここで、時刻ｔ１～ｔ９のステップＳ４１２で、ＳＩＯモデルＨ３０７は、ＳＩＯ端子２３より１ビットずつ通信データを受信する。ここで、受信データ（Ｄ７－Ｄ０）は、「０１０００００１」，すなわち「４１Ｈ」であり、文字コード「Ａ」に相当する。

【０４４２】時刻ｔ１０のステップＳ４１２で、ＳＩＯモデルＨ３０７は、１バイト（８ビット）分のデータを受信し、ストップビットを検出すると、割込Ｉ／Ｆモデル２４に割込み要求信号ＩＲＱを出力するとともに（図３０（ｄ））、ＤＭＡモデルＨ３０５（図２３）にデータ転送要求信号ＤＲＱを出力する（図３０（ｅ））。

【０４４３】ステップＳ４３２（割込み要求有り？）で、ハードウェアモデル２２は、この割込み要求を検知すると、ステップＳ４３３で割込Ｉ／Ｆモデル２４を介してＣＰＵモデル４２に割込み要求信号ＩＲＱを出力する。

【０４４４】ステップＳ３６１（図２７，割込み要求有り？）で、ＣＰＵモデル４２は、割込み要求信号ＩＲＱを検出すると「Ｙ」へ進む。

【０４４５】ステップＳ３６７（割込み受付可？）で、ＣＰＵモデル４２は、割込み受付が可能かを確認する。今回の割込み要求より優先順位の高い割込み処理を実行中であれば、受付不可として「Ｎ」へ進む。ここでは、割込み要求の受付が可能であるとして、「Ｙ」へ進む。

【０４４６】ステップＳ３６８（割込み処理）で、ＣＰＵモデル４２は、定められた割込み処理ルーチンを呼び出す。ここで、割込み処理ルーチンは、ＤＭＡモデルＨ３０５に対して、チャネル０のＤＭＡを許可する命令であるり、アドレス１９００Ｈ番地以降に記述してあるとする。

【０４４７】ＣＰＵモデル４２は、プログラムカウンタに１９００Ｈを格納して、図２７に示すＣＰＵモデルシミュレーションを再帰的に呼び出す。ステップＳ３６１→Ｓ３６２→Ｓ１０１→Ｓ１０２→Ｓ１２３→Ｓ１０４→Ｓ１０５→Ｓ１０６→Ｓ１０７→Ｓ１２８と処理が進む。

【０４４８】時刻ｔ１１～ｔ１４の周辺回路処理サブルーチン（図２８）において、ステップＳ４３１→Ｓ４０３→Ｓ４１０→Ｓ４１１→Ｓ４１２の順に処理が行われ、１９００Ｈ番地の命令に対応する端子信号Ｉ／Ｏ，／ＷＲ，／ＣＳ０，ＡＤＲＳ，ＤＡＴＡなどがハードウェアモデル２２に供給される（図３０（ｈ）～

（ｏ））。

【０４４９】時刻ｔ１４のステップ４１２で、ＤＭＡモデルＨ３０５は、ＤＭＡ転送が可能になる。以上で、１９００Ｈ番地の処理が終了し、ステップＳ４１３→Ｓ４２４→Ｓ４１１と進み、ハードウェアシミュレーション完了信号「ＤＯＮＥ」を、Ｓ４１０→Ｓ４０３→Ｓ１０９（図２７）の順にＣＰＵモデル４２に返す。これにより、ＣＰＵモデル４２は、開始に戻り、次のシミュレーション待ちとなる。

【０４５０】一方、ハードウェアシミュレータ２－２は、ステップＳ２１３→Ｓ２１６と進む。

【０４５１】時刻ｔ１５のステップＳ２１６（外部端子変化？）で、ハードウェアシミュレータ２－２は、ＳＩＯモデルＨ３０７（図２３）からのデータ転送要求信号ＤＲＱを検知すると、ハードウェアモデルシミュレーション（Ｓ４１２）のＤＭＡモデルＨ３０５処理を起動し、ＤＭＡモデルＨ３０５は、バス解放要求信号（ＨＲＱ）を活性化する（図３０（ｆ））。以後、ステップＳ４３２→Ｓ４１７→Ｓ４１８（図２９）と進む。

【０４５２】時刻ｔ１５のステップＳ４１８（バス開放要求）で、ＤＭＡモデルＨ３０５は、バス要求Ｉ／Ｆモデル２５を介してＣＰＵモデル４２にバス解放要求信号ＨＲＱを出力して、ステップＳ４１９に移り、ＣＰＵモデル４２からバス使用許可信号ＨＬＤＡが活性化するのを待つ。

【０４５３】時刻ｔ１５のステップＳ３６２（図２７、バス解放要求有？）で、ＣＰＵモデル４２は、ＨＲＱ信号が活性化していることを検出すると、「Ｙ」へ進む。

【０４５４】ステップＳ３６３（バス解放受付可？）で、ＣＰＵモデル４２は、今回のバス解放要求より優先順位の高い処理を実行中であれば、受付不可として「Ｎ」へ進む。ここでは、バス解放要求の受付が可能であるとして、「Ｙ」へ進む。

【０４５５】時刻ｔ１６のステップＳ３６４（バス使用許可）で、ＣＰＵモデル４２は、バス使用許可信号（ＨＬＤＡ：ホールドアクノリッジ）をバス要求Ｉ／Ｆモデル２５経由でＤＭＡモデルＨ３０５に供給する（図３０（ｇ））。

【０４５６】ステップＳ３６５（バス解放）で、ＣＰＵモデル４２は、バスの解放を行い、ステップＳ３６６（バス開放）で、ＤＭＡモデルＨ３０５がバスを開放するのを待つ。

【０４５７】一方、ステップＳ４１９（バス使用許可？）で、ＤＭＡモデルＨ３０５は、バス使用許可信号ＨＬＤＡを検知すると、「Ｙ」へ進み、ステップＳ４１２に移る。

【０４５８】時刻ｔ１７のステップ４１２で、ＤＭＡモデルＨ３０５は、バス使用権を取得し、ＤＭＡ転送処理を開始する。

【０４５９】まず、ＤＭＡモデルＨ３０５は、端子信号Ｉ／Ｏ，／ＣＳ１，ＡＤＲＳ（ＳＩＯ）がＳＩＯモデルＨ３０７に供給される（図３０（ｈ）（ｌ）（ｎ））。

【０４６０】時刻ｔ１８で、ＤＭＡモデルＨ３０５は、端子信号／ＲＤを活性化し（図３０（ｊ））、ＳＩＯモデルＨ３０７よりＤＡＴＡ「Ａ」を読み込む（図３０（ｏ））。また、ＳＩＯモデルＨ３０７は、データの読み出しを検知すると、データ転送要求信号を非活性化する（図３０（ｅ））。

【０４６１】時刻ｔ１９，ｔ２０で、ＤＭＡモデルＨ３０５は、端子信号を元に戻し、ＳＩＯモデルＨ３０７からの読み出しサイクルを終了する。

【０４６２】時刻ｔ２１で、ＤＭＡモデルＨ３０５は、端子信号／ＭＥＭ，／ＣＳ２，ＡＤＲＳ（２２００Ｈ）をデータメモリモデルＨ３０３に供給する（図３０（ｈ）（ｍ）（ｎ））。

【０４６３】時刻ｔ２２で、ＤＭＡモデルＨ３０５は、端子信号／ＷＲを活性化し（図３０（ｊ））、データメモリモデルＨ３０２の２２００Ｈ番地にＤＡＴＡ「Ａ」を書き込む（図３０（ｏ））。

【０４６４】時刻ｔ２３，ｔ２４で、ＤＭＡモデルＨ３０５は、端子信号を元に戻し、データメモリモデルＨ３０３への書き込みサイクルを終了する。

【０４６５】このように、時刻ｔ２５～ｔ３５の間もＳＩＯ→ＤＭＡ→メモリの順にデータが転送され、通信データが終了するまで、繰り返えされる。

【０４６６】今の場合、ＳＩＯモデルＨ３０７は、受信したデータ「Ａ」，「Ｂ」を読み込み、このデータをデータメモリモデルＨ３０３に順次ＤＭＡ転送する。ＤＭＡモデルＨ３０５は、ＨＤＭＡテーブルＨ３０４の２２００，２２０１番地にそれぞれ「Ａ」，「Ｂ」を記憶する。

【０４６７】時刻ｔ３６のステップＳ４１２（図２９）で、ＤＭＡモデルＨ３０５は、ＤＭＡ転送が終了すると、バス開放要求信号ＨＲＱを非活性化する。以降、ステップＳ４３２→Ｓ４１７→Ｓ４１８（図２９）と進む。

【０４６８】ステップＳ４１８（バス開放要求出力）で、ＤＭＡモデルＨ３０５は、バス解放要求信号（ＨＲＱ）を非活性化する（図３０（ｆ））。ＤＭＡモデルＨ３０５は、バス要求Ｉ／Ｆモデル２５を介してＣＰＵモデル４２にバス解放要求信号ＨＲＱを出力して、ステップＳ４１９に移り、ＣＰＵモデル４２からバス使用許可信号ＨＬＤＡが非活性化するのを待つ。

【０４６９】一方、時刻ｔ３７のステップＳ３６６（図２７、バス解放？）で、ＣＰＵモデル４２は、ＨＲＱ信号が非活性化していることを検出すると、バス使用許可信号ＨＬＤＡを非活性化して、バス要求Ｉ／Ｆモデル２５経由でＤＭＡモデルＨ３０５に供給する（図３０（ｇ））。その後、ＣＰＵモデル４２は、ステップＳ１１０で「Ｙ」へ進み、開始に戻る。

【０４７０】一方、ステップＳ４１９（バス使用許可？）で、ＤＭＡモデルＨ３０５は、非活性化したバス使用許可信号ＨＬＤＡを検知すると、「Ｙ」へ進み、ステップＳ４１２に移る。以上で、ＤＭＡ転送は終了し、ステップＳ４１２→Ｓ４３２→Ｓ４１７→Ｓ４２４に進む。

【０４７１】ステップＳ４２４（仮想モデルデータ更新要？）で、ハードウェアモデル２－２は、ハードウェアモデルシミュレーションＳ４１２の結果、仮想モデル７２側のデータ更新が必要か否かを判断する。ハードウェアモデル２－２は、データメモリモデルＨ３０３のＨＤＭテーブルの内容が書き変わったことを検知し、かつ、機能関数選択テーブル３２（図２５（Ａ））のフラグが「１」であることを確認し、「Ｙ」に進む。

【０４７２】ステップＳ４２５（仮想モデルデータ更新起動）で、ハードウェアモデル２－２は、ＨＤＭテーブルＨ３０４の２２００，２２０１番地に記憶した「Ａ」，「Ｂ」を、仮想モデル４２に渡す。

【０４７３】ステップＳ４２１（図２８）で、仮想モデル７２は、受領した更新データを仮想データメモリモデルＶ４０３に付属するＶＤＭテーブルＶ４０４の２２００，２２０１番地に書き込む。

【０４７４】ステップＳ２１２で、ハードウェアシミュレータ２－２は、ＤＭＡモデルＨ３０５のＤＭＡ転送処理が完了したことを示す「ＤＯＮＥ」をステップＳ４１０→Ｓ４０３→Ｓ１２８経由でＣＰＵモデル４２に返す。

【０４７５】この後、ハードウェアシミュレータ２－２は、ステップＳ４１１でアイドリング状態、即ち、ステップＳ２１３（端子信号変化？）またはＳ２１６（外部端子変化）のいずれかが「Ｙ」になるのを待つ。

【０４７６】《ターゲットプログラムＴ３１１：データメモリモデル読み出し》ＣＰＵモデル４２は、仮想モデル７２の仮想データメモリモデルＶ４０３（図２４）の２２００番地からデータ“Ａ”を読み出す。

【０４７７】《ターゲットプログラムＴ３１２：データ判断》ＣＰＵモデル４２は、読み出したデータが「Ａ」（４１Ｈとする）であるか否かを判断する。「Ａ」であれば、「Ｙ」へ進み、Ｔ３１３の処理に移る。「Ａ」でなければ、「Ｎ」へ進み、Ｔ３１４の処理に移る。

【０４７８】《ターゲットプログラムＴ３１３：ポート出力「１」》ＣＰＵモデル４２は、読み出したデータが「Ａ」であれば、出力ポートに「１」を出力し、正確に受信できたことを表示する。

【０４７９】《ターゲットプログラムＴ３１４：ポート出力「０」》ＣＰＵモデル４２は、読み出したデータが「Ａ」以外であれば、出力ポートに「０」を出力し、正確に受信できなかったことを表示する。

【０４８０】以上で、図２６のターゲットプログラムによる動作説明を終える。

【０４８１】このように、ハードウェアシミュレータ２－２を使ってＤＭＡ転送を行い、データメモリモデルＨ３０３の記憶内容を書き替えても、データ更新機能により、仮想モデル７２側の記憶内容も同時に更新されるので、その後のターゲットプログラムで、仮想モデル７２側の仮想データメモリモデルＶ４０３から記憶内容を読み出しても、正しい処理が実行できる。

【０４８２】つまり、ＤＭＡ転送などＣＰＵモデル４２が関知しない状態で、ハードウェアモデル２２のデータメモリモデルＨ３０３が書き替えられても、仮想モデル７２側の仮想データメモリモデルＶ４０３も同時に更新するようにしたので、ＤＭＡ転送直後のＴ３１１（図２６）で仮想データメモリモデルＶ４０３を読み出しても、ターゲットプログラムＴ３１２で判断を誤ることなく正常に処理される。

【０４８３】また、シミュレーションの処理を仮想モデル７２で行うか、ハードウェアシミュレータ２－２で行うかを判断テーブル内に設けたフラグにより切り替えるようにした。このため、シミュレータ装置の操作者は、テーブルの内容（機能関数や判断条件）を書き換えることなく、フラグの書き換えだけで仮想モデル７とハードウェアモデル２２の選択が容易に行える。従って、操作者は、システムシミュレーションの最中に、シミュレーションの用途に応じて適宜フラグを書き替え、問題のあるハードウェアモデル２２だけを集中して検証することができるようになる。問題が解決したハードウェアモデル２２内のサブモデルは、仮想モデル７２に置き換えてシミュレーションすることができるので、検証に要する時間を大幅に短縮できる。

【０４８４】さらに、外部端子の変化をチェックする手段を設けたので、ＣＰＵモデル４２の端子信号の変化だけでなく、外部端子の信号変化をトリガとして、ハードウェアシミュレータ２－２を起動することができる。この結果、ＳＩＯのような通信モデルがシミュレーションできるようになった。

【０４８５】また、ハードウェアシミュレータ２－２側に、割込みＩ／Ｆモデル２４やバス要求Ｉ／Ｆモデル２５を設けたので、マイコンシミュレータ１側の動作状態に関わらず、要求信号を渡すことができる。その結果、ＤＭＡやＳＩＯのシミュレーションが可能になった。

【０４８６】［第２のケース：仮想モデルシミュレータ７２によるＤＭＡ転送の動作説明］次に、仮想モデルの仮想通信モデルＶ４０５を使って、ＳＩＯデータを受信する例を説明する。この動作の概要は次のようである。すなわち、仮想通信モデルＶ４０５は、シリアル入出力Ｓ＿ＳＩＯから受信データを得る。このデータは仮想データメモリモデルＶ４０３のデータテーブルＶＤＭテーブルＶ４０４に書き込まれる。これと同時に、ハードウェアモデル２２のデータメモリモデルＨ３０３のデータテーブルＨＤＭテーブルＨ３０４のデータも更新され

る。次に、ハードウェアモデル２２のデータメモリモデルＨ３０３のデータテーブルＨＤＭテーブルＨ３０４からデータを読み出す。

【０４８７】以下、図２６のターゲットプログラムに沿って説明する。

【０４８８】《ターゲットプログラムＴ３０１：ＳＩＯ初期設定の処理》ＣＰＵモデル４２は、仮想通信モデルＶ４０５内のＶＲＴテーブルＶ４０６に諸条件を設定する。設定内容はハードウェアシミュレータ２－２による場合と同じである。この設定は、仮想通信モデルＶ４０５に記述されたプログラムに従って、シミュレーション装置のシリアル入出力Ｉ／ＦモデルＳ＿ＳＩＯ（図２）に対して行われ、その後、データ更新（図２８、Ｓ４２３）によりＳＩＯモデルＨ３０７内のＨＳＩ設定レジスタＨ３０８にも書き込まれる。

【０４８９】《ターゲットプログラムＴ３０２：ＤＭＡ初期設定》ＣＰＵモデル４２は、仮想通信モデルＶ４０５内のＶＲＴテーブルＶ４０６に諸条件を設定する。この設定内容もハードウェアシミュレータ２０１による場合と同じである。この設定は、通信モデルＶ４０５に記述されたプログラムに従って、シミュレーション装置のＤＭＡ装置Ｓ＿ＤＭＡ（図２）に対して行われ、その後、データ更新（図２８、Ｓ４２３）によりＤＭＡモデルＨ３０５内のＨＳＩ設定レジスタＨ３０６にも書き込まれる。

【０４９０】《ターゲットプログラムＴ３０３：ＳＩＯ割込み許可》ＣＰＵモデル４２は、仮想通信モデルＶ４０５内のＶＲＴテーブルＶ４０６に割込みの許可を設定する。この設定は、通信モデルＶ４０５に記述されたプログラムに従って、シミュレーション装置のＤＭＡ装置Ｓ＿ＤＭＡ（図２）に対して設定されるが、ＤＭＡモデルＨ３０５には、書き込まれないようにする。これは、仮想モデル７２とハードウェアシミュレータ２－２が同時に通信データを受信することがないようにするためである。

【０４９１】《ターゲットプログラムＴ３０４：割込み発生》シリアル入出力Ｉ／ＦモデルＳ＿ＳＩＯ（図２）は、通信データの最初の１バイトを受信し終わると、割込み制御装置Ｓ＿ＩＮＴ（図２）に対して、割込み要求を行う。割込み制御装置Ｓ＿ＩＮＴは、プロセッサＳ＿ＣＰＵに対して割り込み要求を行い、割込み処理が記述されたアドレスを渡す。プロセッサＳ＿ＣＰＵは、定められた割込み処理ルーチンを呼び出す。ここで、割込み処理ルーチンは、ＤＭＡ装置Ｓ＿ＤＭＡに対して、チャネル０のＤＭＡを許可する命令であるとする。

【０４９２】《ターゲットプログラムＴ３２１：ＤＭＡ許可》図２７で、ステップＳ３６１→Ｓ３６２→Ｓ１０１→…→Ｓ１２８と進み、ＣＰＵモデル４２は、仮想通信モデルＶ４０５のＶＲＴテーブルＶ４０６（図２４）にＤＭＡ許可を設定するとともに、仮想通信モデルＶ４０５は、記述されたプログラムに従って、ＤＭＡ装置Ｓ＿ＤＭＡ（図２）にチャネル０のＤＭＡを許可する。

【０４９３】以後のＳＩＯデータ受信とＤＭＡ転送の処理は、仮想モデルシミュレーションＳ４２２やハードウェアモデルシミュレーションＳ４１２（図２８）に係わらず行われる。即ち、ＤＭＡ装置Ｓ＿ＤＭＡ（図２）は、シリアル入出力Ｉ／ＦモデルＳ＿ＳＩＯから受信データ「Ａ」、「Ｂ」を読み出し、仮想データメモリモデルＶ４０３内のＶＤＭテーブルＶ４０４の所定の領域（２２００Ｈ，２２０１番地）に対応するメモリＳ＿ＲＡＭの領域にＤＭＡ転送する。

【０４９４】《ターゲットプログラムＴ３２２：ＤＡＭ終了？》ステップＳ４２２（図２８、仮想モデルシミュレーション）で、仮想通信モデルＶ４０５（図２４）は、ＤＭＡ転送が終了したことを検知すると、ステップＳ４２０の処理に移る。

【０４９５】ステップＳ４２２（ハードウェアデータ更新起動）で、仮想モデル７２は、ハードウェアモデル２２に仮想データメモリモデルＶ４０３内のＶＤＭテーブルＶ４０４の更新データ「２２００，Ａ」、「２２０１，Ｂ」を渡す。

【０４９６】ステップＳ４２３（データ更新）で、ハードウェアモデル２２は、受領した更新データ「Ａ」、「Ｂ」をデータメモリモデルＨ３０３内のＨＤＭテーブルＨ３０４の２２００Ｈ，２２０１Ｈ番地に書き込む。

【０４９７】以上で、図２８の周辺回路処理サブルーチンを終了し、図２７のステップＳ１２８に戻り、Ｓ１０９→Ｓ１１０と進み、開始に戻る。

【０４９８】以上で、割込み処理ルーチンを終了し、割込みが発生したターゲットプログラムのステップに戻る。

【０４９９】《ターゲットプログラムＴ３１１：データメモリモデル読み出し》ＣＰＵモデル４２は、ハードウェアモデル２２のデータメモリモデルＨ３０３（図２３）の２２００番地からデータ“Ａ”を読み出す。

【０５００】《ターゲットプログラムＴ３１２：データ判断》ＣＰＵモデル４２は、読み出したデータが「Ａ」（４１Ｈとする）であるか否かを判断する。「Ａ」であれば、「Ｙ」へ進み、Ｔ３１３の処理に移る。「Ａ」でなければ、「Ｎ」へ進み、Ｔ３１４の処理に移る。

【０５０１】《ターゲットプログラムＴ３１３：ポート出力「１」》ＣＰＵモデル４２は、読み出したデータが「Ａ」であれば、出力ポートに「１」を出力し、正確に受信できたことを表示する。

【０５０２】《ターゲットプログラムＴ３１４：ポート出力「０」》ＣＰＵモデル４２は、読み出したデータが「Ａ」以外であれば、出力ポートに「０」を出力し、正確に受信できなかったことを表示する。

【０５０３】以上で、図２６のターゲットプログラムの検証を終了する。

【０５０４】ハードウェアシミュレータ２－２によれば、図３４のタイミングチャートから明らかなように、ＳＩＯ端子２３に１バイトのデータが入力され、ＤＭＡ転送によりデータメモリモデルＨ３０３に書き込みが終了するまでに３０クロック（図３０（ａ）マシンサイクル）程度必要となるが、ハードウェアシミュレータ２－２では、シミュレーションの規模が大きいので、１クロック分を演算するのに、約１０ｍＳを要する。したがって、１バイト受信には、３００ｍＳもの時間が必要となることになる。このため、ＳＩＯモデルＨ３０７のデータ転送をハードウェアモデル２２では、３ｂｐｓ程度の通信速度はシミュレーションできるが、最近の９６Ｋｂｐｓ等の高速モデムを、実使用状態でシミュレーションすることは不可能である。

【０５０５】これに対し、仮想通信モデルＶ４０５を使用することで、高速モデムであっても、リアルタイムにシミュレーションすることが可能になる。

【０５０６】また、仮想通信モデルＶ４０５で受信したデータは、ハードウェアモデル２２にも書き込まれるので、ターゲットプログラムのＴ３１１，Ｔ３１２（図２６）をハードウェアモデル２２で実行しても、正しい処理が実行される。

【０５０７】ハードウェアシミュレータ２０１で、１度、ＳＩＯモデルＨ３０７やＤＭＡモデルＨ３０５が正しく動作することが確認できれば、その後はハードウェアシミュレーションをする必要はなく、これ以降は、機能関数選択テーブル（図２５）のフラグを「０」→「１」に書き替えるだけで、仮想通信モデルＶ４０５を使ってターゲットプログラムを検証することができるので、上述の効果を得ることができるようになる。

【０５０８】なお、第１～第３実施例では、仮想モデルをマイコンシミュレータの一部として説明したが、これに限定されるもにではなく、ハードウェアシミュレータ側に設けてもよいし、マイコンシミュレータやハードウェアシミュレータとは別に構成してもよい。

【０５０９】また、仮想モデルやハードウェアモデルを構成する周辺回路、あるいは、機能関数、機能関数選択テーブルなどは、第１～第３実施例に限定されるもにではなく、電子機器を構成する周辺回路によって適宜変更しうるものである。

【０５１０】また、シミュレーションを実行するときに、周辺回路の入出力状態をログファイルに記録しておき、次にシミュレーションするときに、ログファイルの内容を仮想モデルやハードウェアモデルの入出力情報として供給することで、シミュレーションに要する時間を短縮することができる。

【０５１１】

【発明の効果】本発明によれば、電子機器を構成するハードウェアの検証を、電子機器に組み込むプログラムの検証と同時に実行し、かつハードウェアの検証をマイコンシミュレーションで代替する比重を高めたことにより、検証時間の短縮と、検証内容の充実を実現できるという効果がある。

【０５１２】より具体的な効果については、各実施例の説明の末尾に記載したので重複を回避するため省略する。

【図面の簡単な説明】

【図１】本発明の第１実施例のシステムシミュレータの機能ブロック図

【図２】本発明のシステムシミュレーションの母体装置たるシミュレーション装置のブロック図

【図３】図１におけるハードウェアモデル２０の構成例を示すブロック図

【図４】図３のハードウェアモデル２０に対応する仮想モデル７の構成図

【図５】図１における仮想プログラムメモリモデルＶ７１内のテーブル例を示す表

【図６】図１における機能関数選択テーブル３の例を示す表

【図７】図１における機能関数Ｉ／Ｆモデル５における仮想関数変換テーブルとその戻り値を示す表

【図８】本発明の第１実施例および第２実施例におけるターゲットプログラムのリスト例を示す表

【図９】本発明の第１実施例および第２実施例におけるＣＰＵモデルシミュレーションのフローチャート

【図１０】図９における周辺回路処理サブルーチンＳ１０３，Ｓ１０８のフローチャート

【図１１】図１０における仮想モデルシミュレーションＳ２０２のフローチャート

【図１２】図１におけるバスＩ／Ｆモデル６がハードウェアシミュレータ２にデータを書き込むときのタイミングチャート

【図１３】本発明の第２実施例のシステムシミュレータ２－１の機能ブロック図

【図１４】図１３におけるハードウェアモデル２１の構成図

【図１５】図１３における仮想モデル７１の構成図

【図１６】図１３における機能関数選択テーブル３１の例を示す表

【図１７】第２実施例における周辺回路処理サブルーチンの詳細フローチャート。実線は処理の流れ、点線はデータの流れを示す。

【図１８】第２実施例における仮想モデルシミュレーションの詳細フローチャート

【図１９】図１７のうち１点鎖線で示された部分のフローチャート

【図２０】図１７におけるデータ更新Ｓ４２３のフローチャート

【図２１】図１７におけるデータ更新Ｓ４２１のフローチャート

【図２２】本発明の第３実施例のシステムシミュレータ２－２の機能ブロック図
【図２３】図２２におけるハードウェアモデル２２の構成図
【図２４】図２２における仮想モデル７２の構成図
【図２５】図２２における機能関数選択テーブル３２の例を示す表
【図２６】第３実施例におけるターゲットプログラムのフローチャート
【図２７】第３実施例におけるＣＰＵモデルシミュレーションのフローチャート
【図２８】第３実施例における周辺回路処理サブルーチンＳ１２３，Ｓ１２８間のフローチャート
【図２９】第３実施例において外部端子変化により起動されるハードウェアシミュレーションのフローチャート
【図３０】第３実施例におけるシリアル入出力およびＤＭＡ転送時のタイミングチャート
【符号の説明】
１　マイコンシミュレータ
２，２－１，２－２　ハードウェアシミュレータ
３，３１，３２　機能関数選択テーブル
４　ＣＰＵモデル
５　機能関数Ｉ／Ｆモデル
６　バスＩ／Ｆモデル
７，７０，７２　仮想モデル
８　システムＩ／Ｆモデル
９　データ更新Ｉ／Ｆモデル
１０，Ｈ３０１　プログラムメモリモデル
１１，Ｈ３０３　データメモリモデル
１２　Ｉ／ＯポートＡモデル
１３　キーマトリクスモデル
１４　Ｉ／ＯポートＢモデル
１５，Ｈ３１２　デコーダモデル
１６　Ｉ／ＯポートＣモデル
１７　時計モデル
１８　ＧＵＩ
１９　キー入力モデル
２０，２１，２２　ハードウェアモデル
２３　ＳＩＯ端子
２４　割込Ｉ／Ｆモデル
２５　バス要求Ｉ／Ｆモデル
１８１　キー入力ＧＵＩ
１８２　ＬＥＤ表示ＧＵＩ
ＰＣ　プログラムカウンタ
Ｓ＿ＣＰＵ　プロセッサ
Ｓ＿ＲＡＭ　メモリ
Ｓ＿ＲＯＭ　読み出し専用メモリ
Ｓ＿ＩＮＴ　割込制御装置
Ｓ＿ＤＭＡ　ＤＭＡ制御装置
Ｓ＿ＤＳＰ　表示制御装置
Ｓ＿ＩＮＰ　入力部
Ｓ＿ＳＩＯ　シリアル入出力装置
Ｓ＿ＨＤＤ　ハードディスク
ＥＷＳ　ワークステーション
Ｖ７１，Ｖ４０１　仮想プログラムメモリモデル
Ｖ７２　仮想時計モデル
Ｖ７３　仮想キー入力モデル
Ｖ７４　仮想ＬＥＤ表示モデル
Ｖ４０２，Ｈ３０２　ＶＰＭテーブル
Ｖ４０３　仮想データメモリモデル
Ｖ４０４，Ｈ３０４　ＶＤＭテーブル
Ｖ４０５　仮想通信モデル
Ｖ４０６　ＶＲＴテーブル
Ｖ４０７　仮想入力ポートモデル
Ｖ４０８　ＶＩＰ入力レジスタ
Ｖ４０９　仮想キーデコーダモデル
Ｖ４１０　ＶＫＤ出力レジスタ
Ｈ３０５　ＤＭＡモデル
Ｈ３０６　ＨＤＡ設定レジスタ
Ｈ３０７　ＳＩＯモデル
Ｈ３０８　ＨＳＩ設定レジスタ
Ｈ３０９　入力ポートモデル
Ｈ３１０　ＨＩＰ設定レジスタ
Ｈ３１１　出力ポートモデル
Ｈ３１３　ＨＫＤ出力レジスタ

【図６】

機能関数選択テーブル３の例

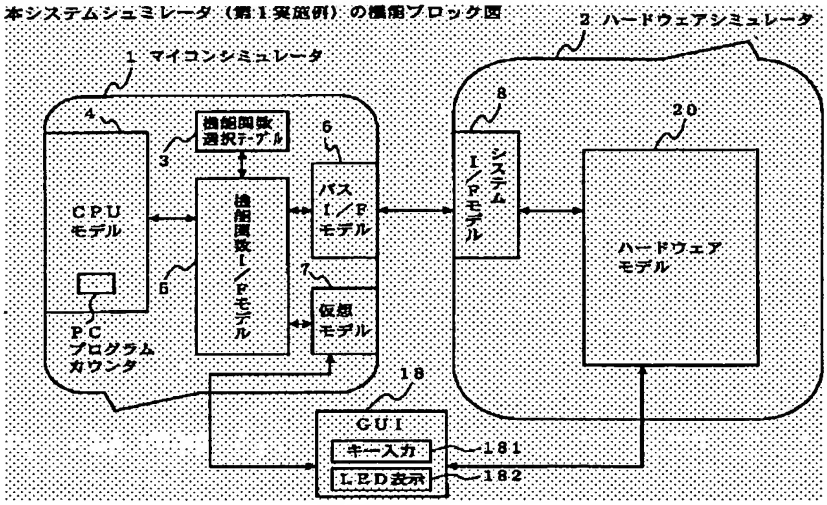
機能関数	引数の条件
FETCH	(1000H≦ADRS＜2000H) OR KEYIN
IORD	(ADRS＝READTIME) & I/O & RD
IOWR	(ADRS＝PORTC) & D* & I/O & WR
CALL	(ADRS＝KEYIN) & (05≦TIME≦15H)

【図７】

仮想関数変換テーブルとその戻り値

変換元	変換後	戻り値
FETCH (m)	MEMm (mは、1000H－1FFFH)	命令コード
IORD (READTIME)	R_TM	時刻情報
IOWR (PORTC, 00)	LMP00	完了コード
IOWR (PORTC, 01)	LMP01	〃
CALL (KEYIN)	K_IN	キーコード

【図１】

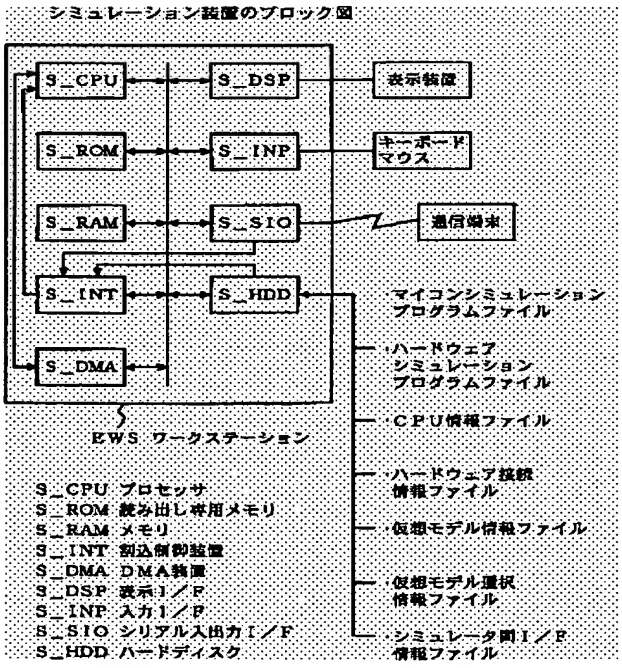


【図５】

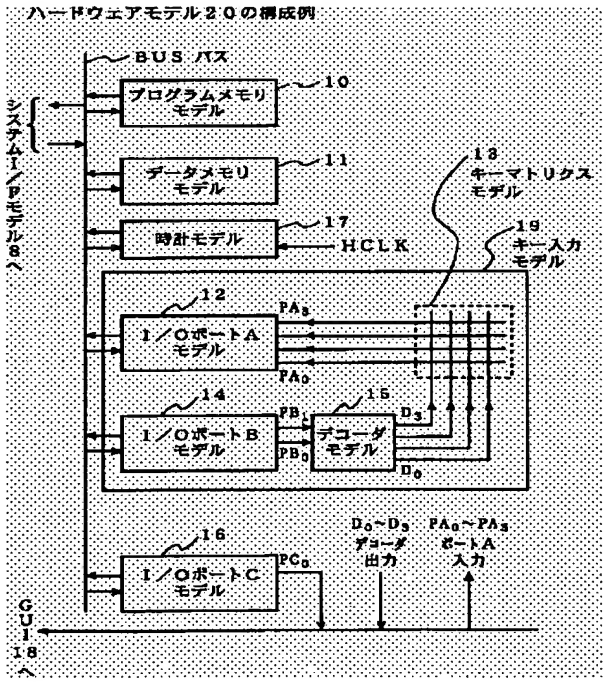
仮想プログラムメモリモデルV７１内のテーブル例

シンボル	戻り値
MEM1001	A201
MEM1002	C501
MEM1003	B204
MEM1004	7406
MEM1005	2002
MEM1006	5AB5
MEM1007	B310
MEM1008	740A
MEM1009	2006
MEM100A	C701
MEM100B	5A03
MEM100C	B2EC
MEM100D	740F
MEM100R	200B
MEM100F	C700

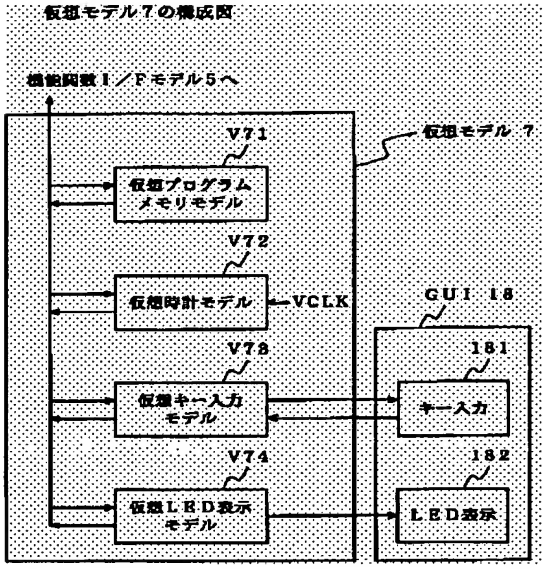
【図２】



【図３】



【図４】

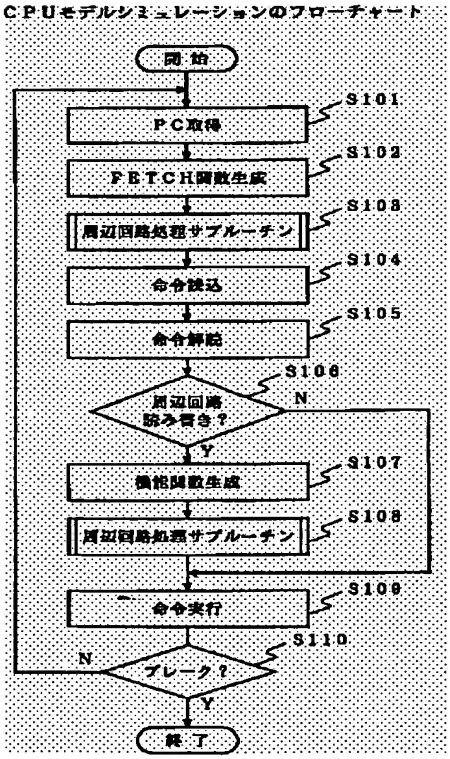


【図８】

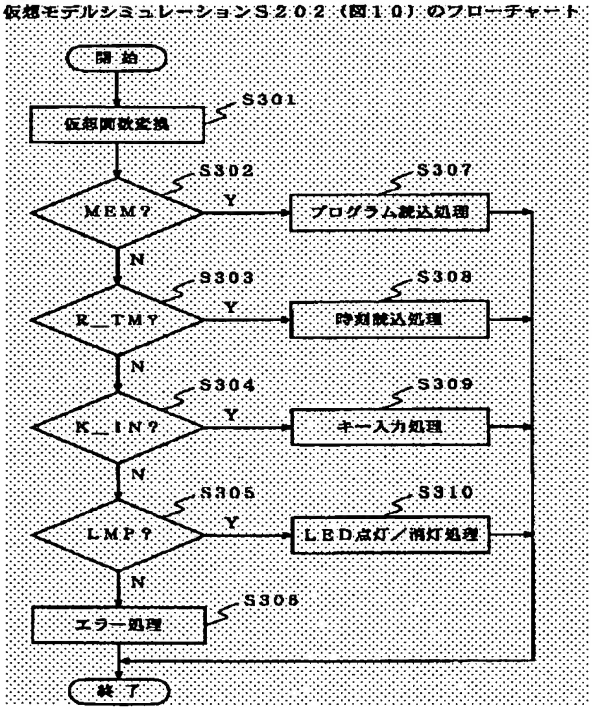
ターゲット・プログラムリスト例

アドレス	命令コード	ニーモニック表示
１００１	Ａ２０１	ＭＯＶ　ＰＯＲＴＢ，＃０１Ｈ
１００２	Ｃ５０１	ＭＯＶ　Ａ，ＰＯＲＴＡ
１００３	Ｂ２０４	ＳＵＢ　Ａ，＃０４Ｈ
１００４	７４０６	ＢＺ　！１００６Ｈ
１００５	２００２	ＢＲ　！１００２Ｈ
１００６	５ＡＢ５	ＭＯＶ　Ａ，ＲＥＡＤＴＩＭＥ
１００７	Ｂ３１０	ＳＵＢ　Ａ，＃１０Ｈ
１００８	７４０Ａ	ＢＺ　！１００ＡＨ
１００９	２００６	ＢＲ　！１００６Ｈ
１００Ａ	Ｃ７０１	ＭＯＶ　ＰＯＲＴＣ，＃０１Ｈ
１００Ｂ	５Ａ０３	ＣＡＬＬ　ＫＥＹＩＮ
１００Ｃ	Ｂ２ＥＣ	ＳＵＢ　Ａ，＃ＥＣＨ
１００Ｄ	７４０Ｆ	ＢＺ　！１００ＦＨ
１００Ｅ	２００Ｂ	ＢＲ　！１００ＢＨ
１００Ｆ	Ｃ７００	ＭＯＶ　ＰＯＲＴＣ，＃００Ｈ
……	……	……
		ＫＥＹＩＮ
３００１	Ａ２０１	ＭＯＶ　ＰＯＲＴＢ，＃０１Ｈ
３００２	Ｃ５０１	ＭＯＶ　Ａ，ＰＯＲＴＡ
……	……	……
３００Ａ	７４０６	ＢＺ　！３００ＣＨ
３００Ｂ	２００２	ＢＲ　！３００２Ｈ
……	……	……
３０１Ｆ	ＣＢＣ９	ＲＥＴ

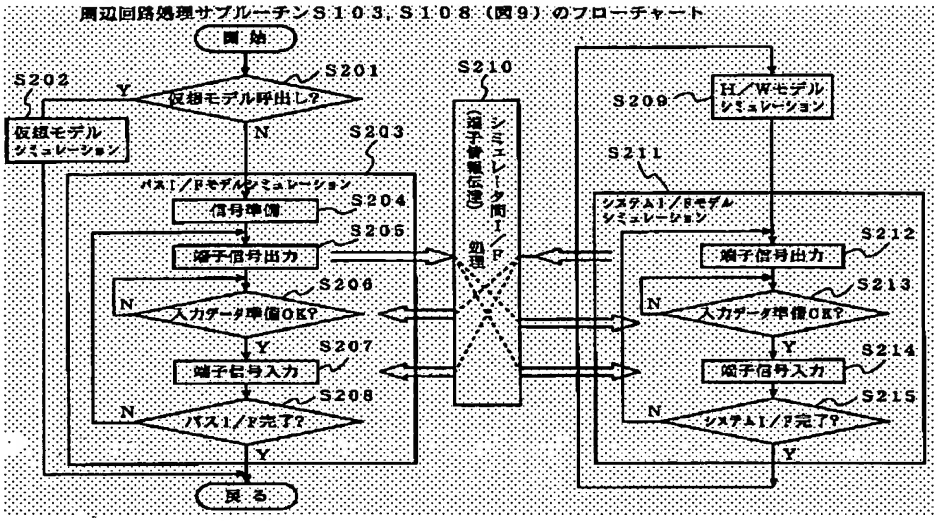
【図９】



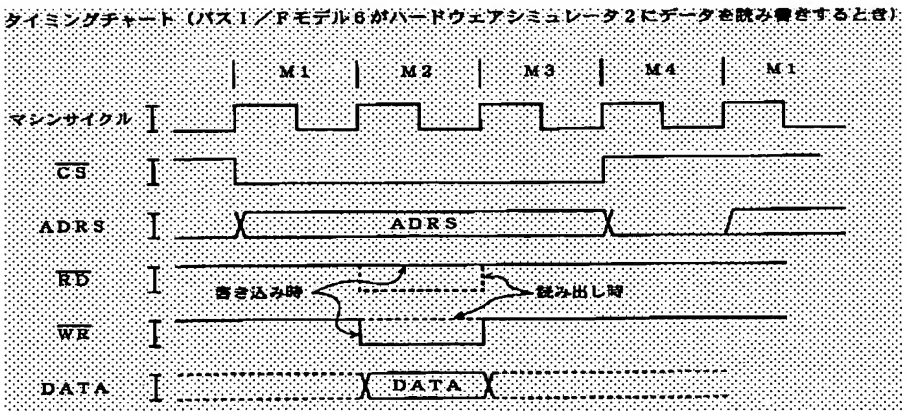
【図１１】



【図１０】



【図１２】

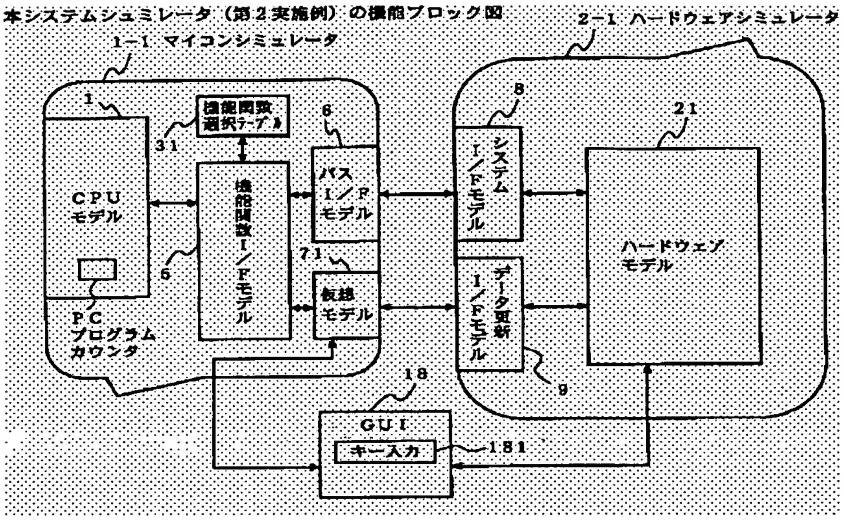


【図１６】

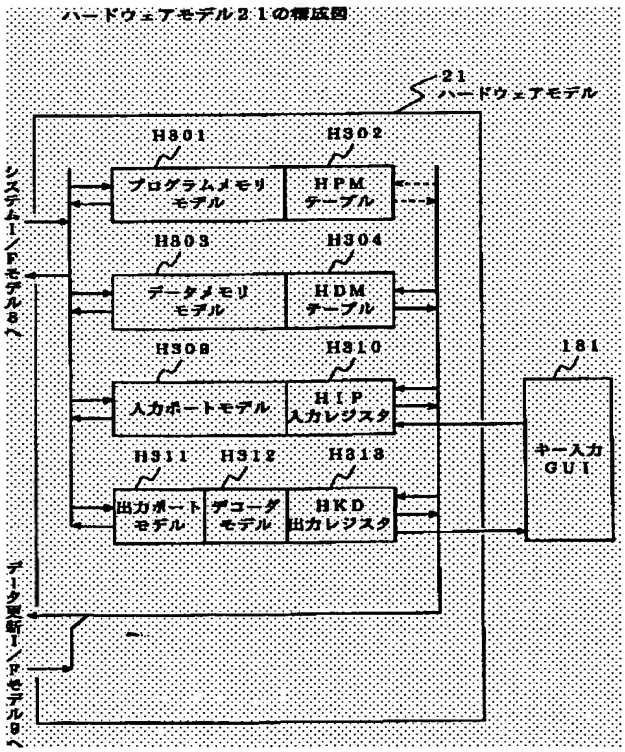
機能関数選択テーブル３１の例

機能関数	引数の条件
FETCH	(１０００H≦ADRS＜２０００H)
IOWR	(ADRS－PORTB)＆ D*＆ I/O ＆ WR

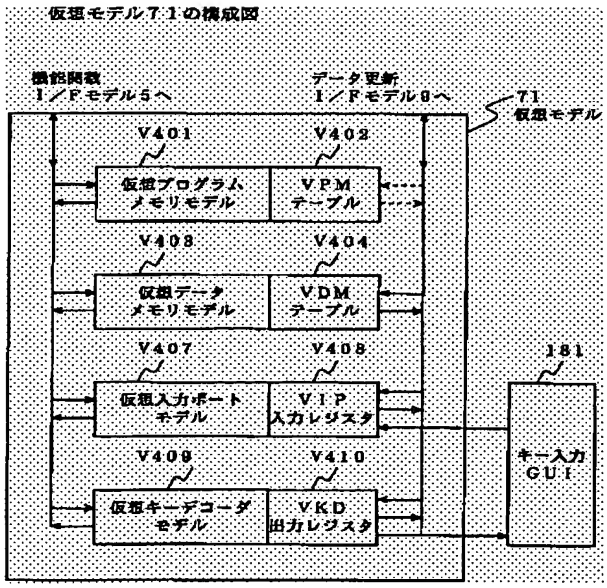
【図１３】


本システムシュミレータ（第２実施例）の機能ブロック図
1-1 マイコンシミュレータ
2-1 ハードウェアシミュレータ
CPU モデル
PC プログラム カウンタ
機能関数 選択テーブル
機能関数 I／Fモデル
バス I／P モデル
仮想 モデル
システム I／Fモデル
データ更新 I／Fモデル
ハードウェア モデル
GUI
キー入力
1
31
5
6
71
8
9
21
18
181

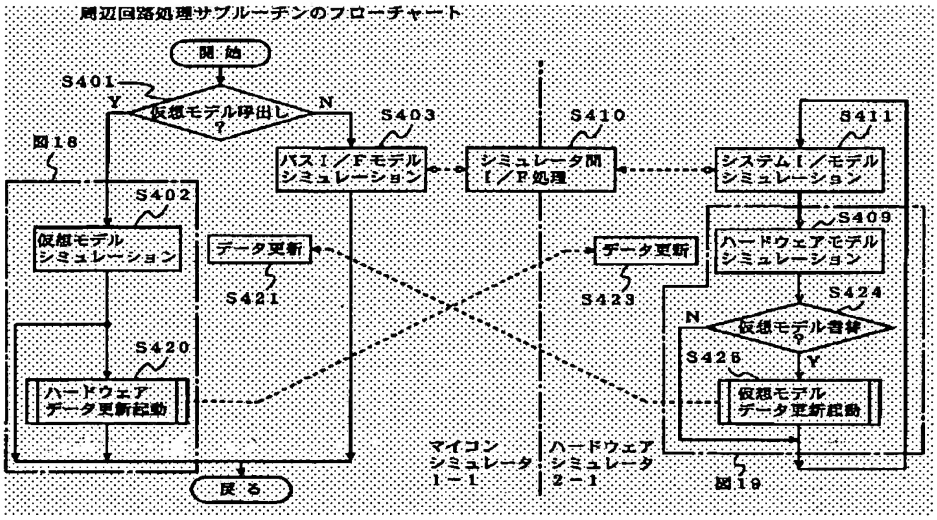
【図１４】


ハードウェアモデル２１の構成図
21 ハードウェアモデル
H301 プログラムメモリ モデル
H302 HPM テーブル
H303 データメモリ モデル
H304 HDM テーブル
H309 入力ポートモデル
H310 HIP 入力レジスタ
H311 出力ポート モデル
H312 デコーダ モデル
H313 HKD 出力レジスタ
181 キー入力 GUI
システムI／Fモデル5へ
データ更新I／Fモデル9へ

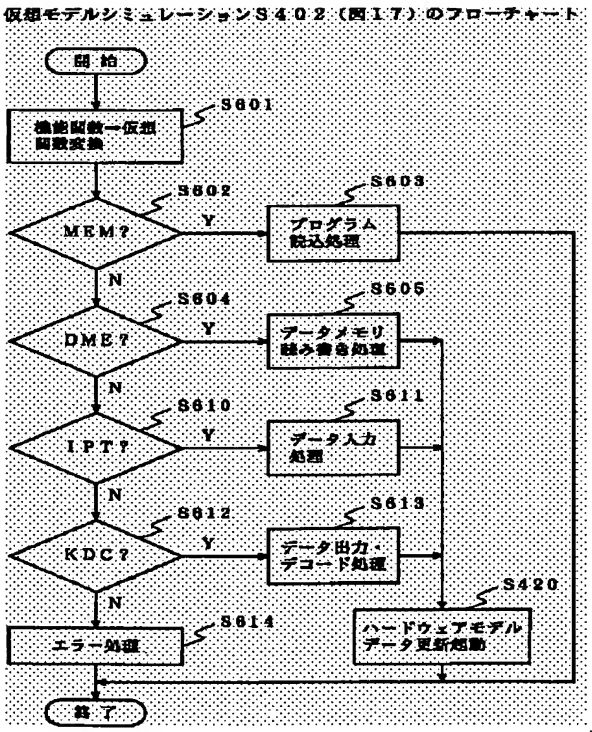
【図１５】


仮想モデル７１の構成図
機能関数 I／Fモデル5へ
データ更新 I／Fモデル9へ
71 仮想モデル
V401 仮想プログラム メモリモデル
V402 VPM テーブル
V403 仮想データ メモリモデル
V404 VDM テーブル
V407 仮想入力ポート モデル
V408 VIP 入力レジスタ
V409 仮想キーデコーダ モデル
V410 VKD 出力レジスタ
181 キー入力 GUI

【図１７】


周辺回路処理サブルーチンのフローチャート
開始
S401
仮想モデル呼出し？
Y
N
S403
バスＩ／Ｆモデルシミュレーション
S410
シミュレータ間Ｉ／Ｆ処理
S411
システムＩ／Ｏモデルシミュレーション
図18
S402
仮想モデルシミュレーション
データ更新
S421
データ更新
S423
S409
ハードウェアモデルシミュレーション
S424
仮想モデル書換？
N
Y
S426
仮想モデルデータ更新起動
S420
ハードウェアデータ更新起動
マイコンシミュレータ１－１
ハードウェアシミュレータ２－１
図19
戻る

【図１８】


仮想モデルシミュレーションＳ４０２（図１７）のフローチャート
開始
機能関数→仮想関数変換
S601
S602
MEM？
Y
S603
プログラム読込処理
N
S604
DME？
Y
S605
データメモリ読み書き処理
N
S610
IPT？
Y
S611
データ入力処理
N
S612
KDC？
Y
S613
データ出力・デコード処理
N
S614
エラー処理
S420
ハードウェアモデルデータ更新起動
終了

【図１９】

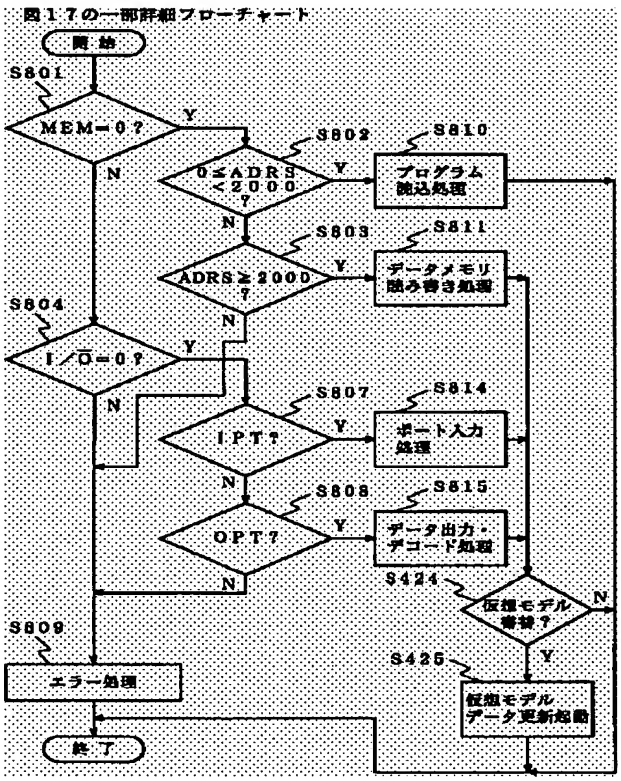
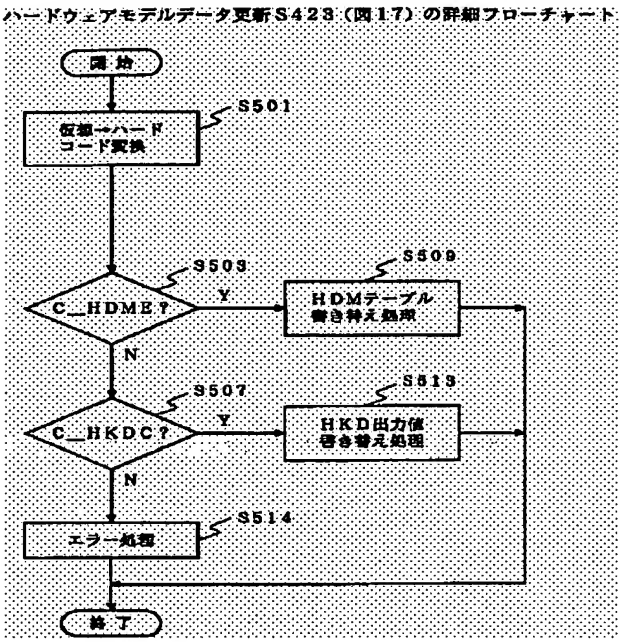
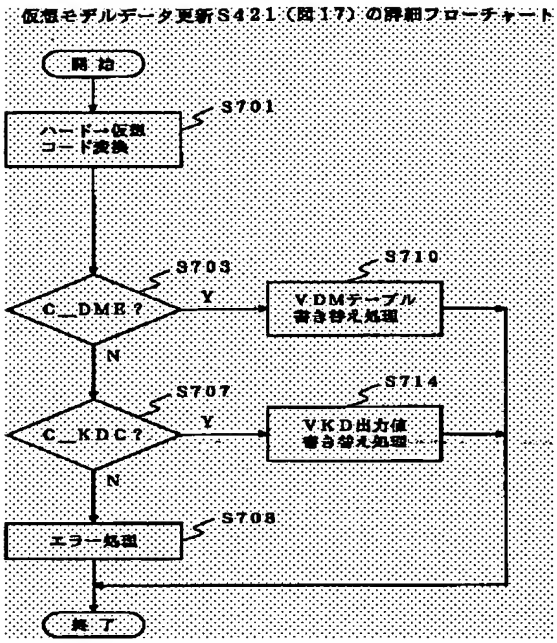


図１７の一部詳細フローチャート
開始
S801
MEM＝０？
Y
N
S802
０≦ADRS＜２０００？
Y
S810
プログラム読込処理
N
S803
ADRS≧２０００？
Y
S811
データメモリ読み書き処理
N
S804
Ｉ／Ｏ＝０？
Y
N
S807
IPT？
Y
S814
ポート入力処理
N
S808
OPT？
Y
S815
データ出力・デコード処理
N
S424
仮想モデル書換？
N
Y
S425
仮想モデルデータ更新起動
S809
エラー処理
終了

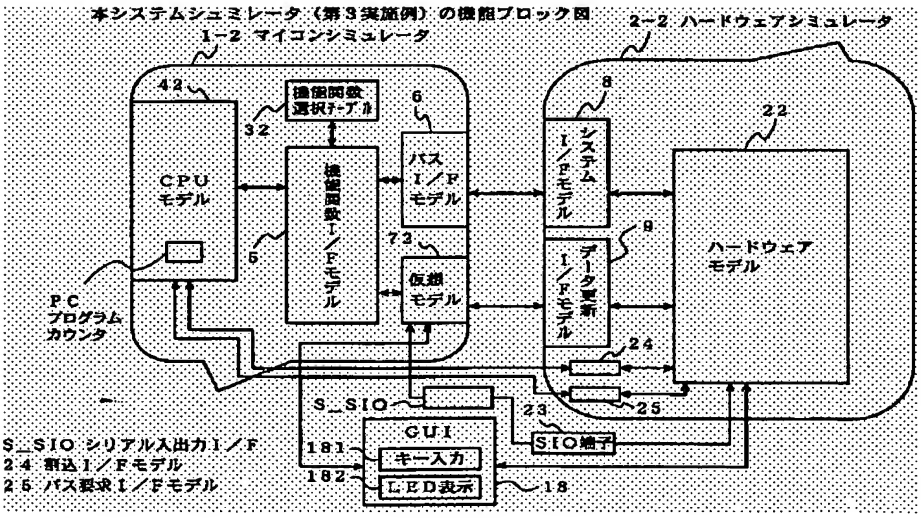
【図２０】


ハードウェアモデルデータ更新S423（図17）の詳細フローチャート
開始
仮想→ハード コード変換
S501
C_HDME?
S503
Y
HDMテーブル 書き替え処理
S509
N
C_HKDC?
S507
Y
HKD出力値 書き替え処理
S513
N
エラー処理
S514
終了

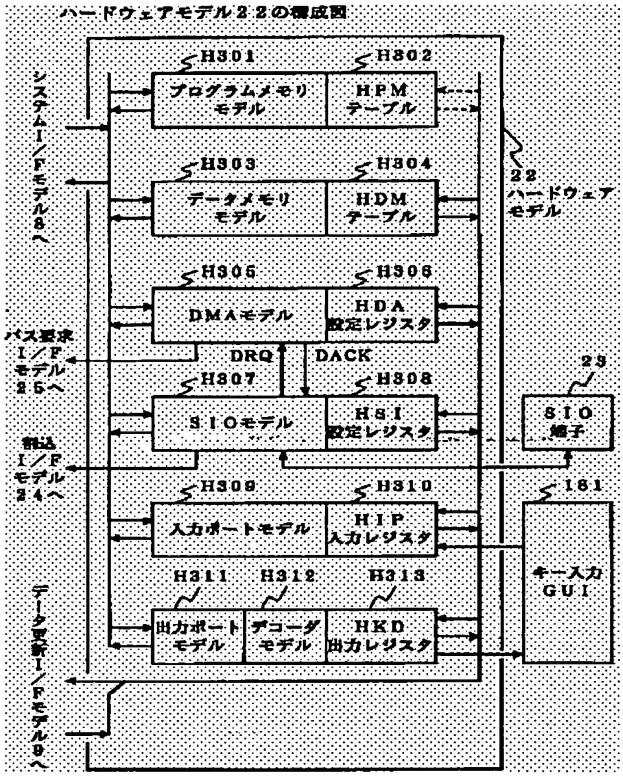
【図２１】


仮想モデルデータ更新S421（図17）の詳細フローチャート
開始
ハード→仮想 コード変換
S701
C_DME?
S703
Y
VDMテーブル 書き替え処理
S710
N
C_KDC?
S707
Y
VKD出力値 書き替え処理
S714
N
エラー処理
S708
終了

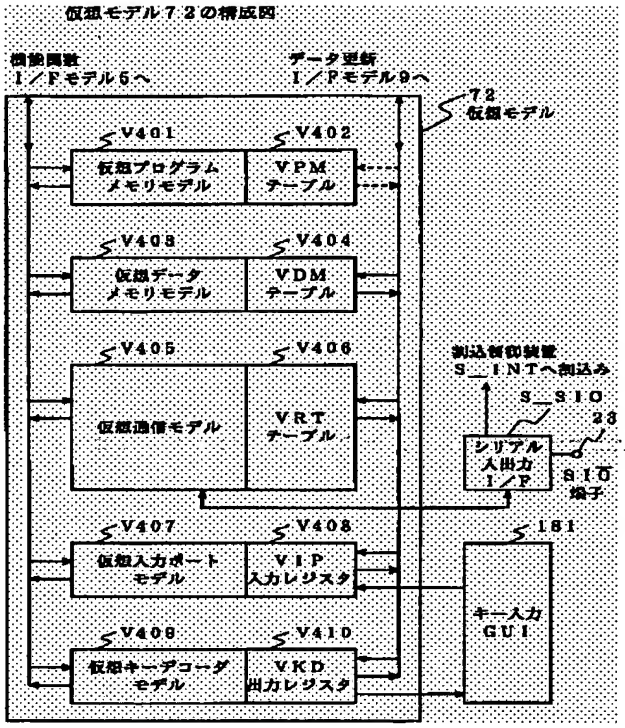
【図２２】


本システムシュミレータ（第３実施例）の機能ブロック図
1-2 マイコンシミュレータ
2-2 ハードウェアシミュレータ
CPU モデル
42
機能関数選択テーブル
32
機能関数I/Fモデル
5
バスI/Fモデル
6
仮想モデル
72
PC プログラムカウンタ
システムI/Fモデル
8
データ更新I/Fモデル
9
ハードウェアモデル
22
24
25
23
S_SIO
SIO端子
GUI
181
キー入力
182
LED表示
18
S_SIO シリアル入出力I/F
24 割込I/Fモデル
25 バス要求I/Fモデル

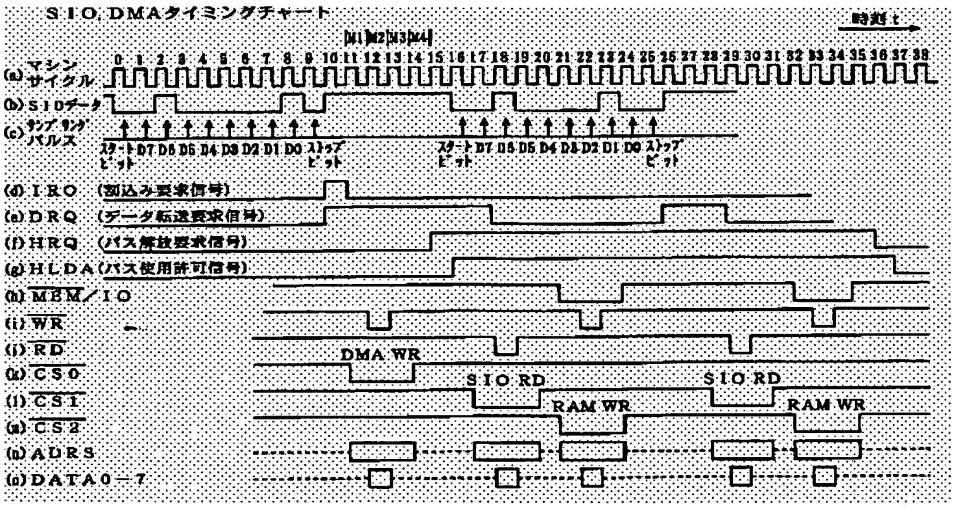
【図２３】


ハードウェアモデル２２の構成図
システムＩ／Ｆモデル８へ
H301
プログラムメモリモデル
H302
HPMテーブル
H303
データメモリモデル
H304
HDMテーブル
22
ハードウェアモデル
H305
DMAモデル
H306
HDA設定レジスタ
バス要求Ｉ／Ｆモデル２５へ
DRQ
DACK
H307
ＳＩＯモデル
H308
HSI設定レジスタ
23
ＳＩＯ端子
割込Ｉ／Ｆモデル２４へ
H309
入力ポートモデル
H310
HIP入力レジスタ
161
キー入力ＧＵＩ
H311
出力ポートモデル
H312
デコーダモデル
H313
HKD出力レジスタ
データ更新Ｉ／Ｆモデル９へ

【図２４】


仮想モデル７２の構成図
機能関数Ｉ／Ｆモデル５へ
データ更新Ｉ／Ｆモデル９へ
72
仮想モデル
V401
仮想プログラムメモリモデル
V402
VPMテーブル
V403
仮想データメモリモデル
V404
VDMテーブル
V405
仮想通信モデル
V406
VRTテーブル
割込制御装置
S_INTへ割込み
S_SIO
23
シリアル入出力Ｉ／Ｆ
ＳＩＯ端子
V407
仮想入力ポートモデル
V408
VIP入力レジスタ
181
キー入力ＧＵＩ
V409
仮想キーデコーダモデル
V410
VKD出力レジスタ

【図３０】


ＳＩＯ,DMAタイミングチャート
時刻ｔ
T1 T2 T3 T4
(a)マシンサイクル
(b)ＳＩＯデータ
(c)サンプリングパルス
スタートビット D7 D6 D5 D4 D3 D2 D1 D0 ストップビット
(d)ＩＲＱ (割込み要求信号)
(e)ＤＲＱ (データ転送要求信号)
(f)ＨＲＱ (バス解放要求信号)
(g)ＨＬＤＡ (バス使用許可信号)
(h)M/IO
(i)WR
(j)RD
(k)CS0
(l)CS1
(m)CS2
(n)ADRS
(o)DATA0－7
DMA WR
SIO RD
RAM WR
SIO RD
RAM WR

【図２５】

機能関数選択テーブル３２の例

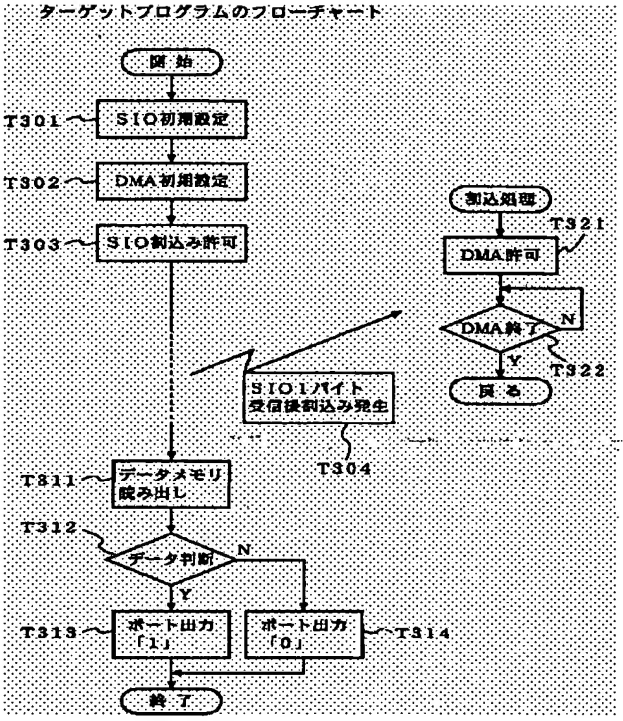
（Ａ）【ハードウエアモデルでＳＩＯ、ＤＭＡ転送を行う場合】

選択可能な関数	引数の条件	フラグ
ＦＥＴＣＨ	（１０００Ｈ≦ＡＤＲＳ＜２０００Ｈ）＆ＭＥＭ ＆ ＲＤ	１
ＭＥＭ	ＡＤＲＳ≧２０００Ｈ＆ＭＥＭ ＆（ＲＤ ｏｒ ＷＲ）	１
ＩＯＲＤ	（ＡＤＲＳ＝ＰＯＲＴＡ）＆Ｉ／Ｏ ＆ ＲＤ （ＡＤＲＳ＝ＰＯＲＴＢ）＆Ｉ／Ｏ ＆ ＲＤ （ＡＤＲＳ＝ＤＭＡ）＆Ｉ／Ｏ ＆ ＲＤ （ＡＤＲＳ＝ＳＩＯ）＆Ｉ／Ｏ ＆ ＲＤ	０ ０ ０ ０
ＩＯＷＲ	（ＡＤＲＳ＝ＰＯＲＴＡ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ （ＡＤＲＳ＝ＰＯＲＴＢ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ （ＡＤＲＳ＝ＤＭＡ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ （ＡＤＲＳ＝ＳＩＯ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ	０ ０ ０ ０

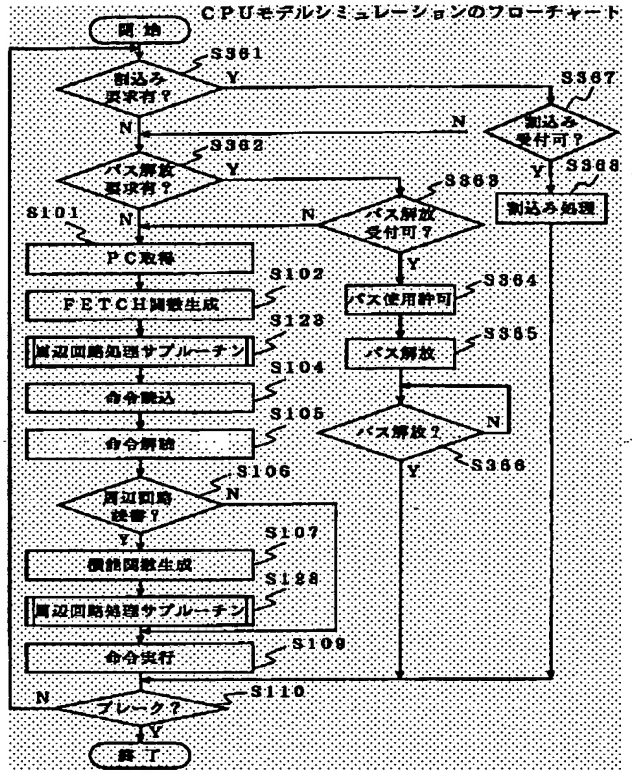
（Ｂ）【仮想モデルでＳＩＯ、ＤＭＡ転送を行う場合】

選択可能な関数	引数の条件	フラグ
ＦＥＴＣＨ	（１０００Ｈ≦ＡＤＲＳ＜２０００Ｈ）＆ＭＥＭ ＆ ＲＤ	１
ＭＥＭ	ＡＤＲＳ≧２０００Ｈ＆ＭＥＭ ＆（ＲＤ ｏｒ ＷＲ）	０
ＩＯＲＤ	（ＡＤＲＳ＝ＰＯＲＴＡ）＆Ｉ／Ｏ ＆ ＲＤ （ＡＤＲＳ＝ＰＯＲＴＢ）＆Ｉ／Ｏ ＆ ＲＤ （ＡＤＲＳ＝ＤＭＡ）＆Ｉ／Ｏ ＆ ＲＤ （ＡＤＲＳ＝ＳＩＯ）＆Ｉ／Ｏ ＆ ＲＤ	０ ０ １ １
ＩＯＷＲ	（ＡＤＲＳ＝ＰＯＲＴＡ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ （ＡＤＲＳ＝ＰＯＲＴＢ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ （ＡＤＲＳ＝ＤＭＡ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ （ＡＤＲＳ＝ＳＩＯ）＆Ｄ＊ Ｉ／Ｏ ＆ ＷＲ	０ ０ １ １

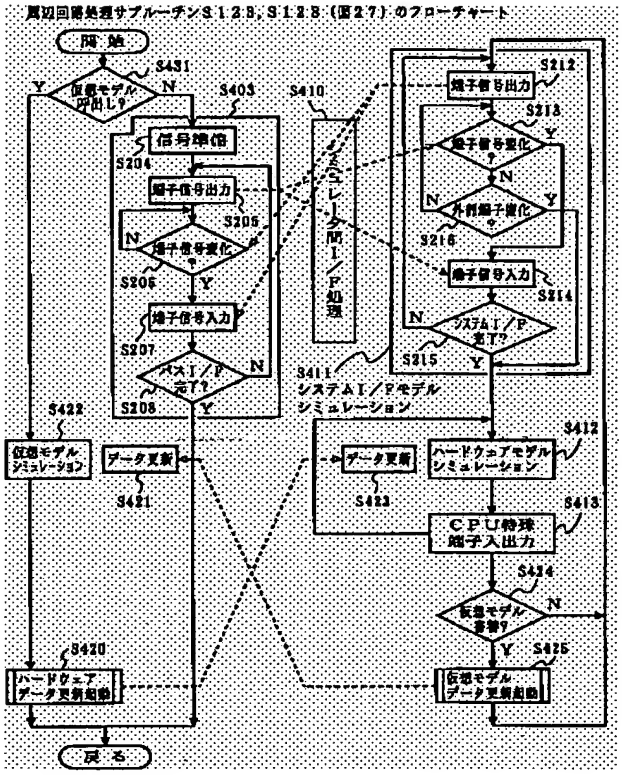
【図２６】



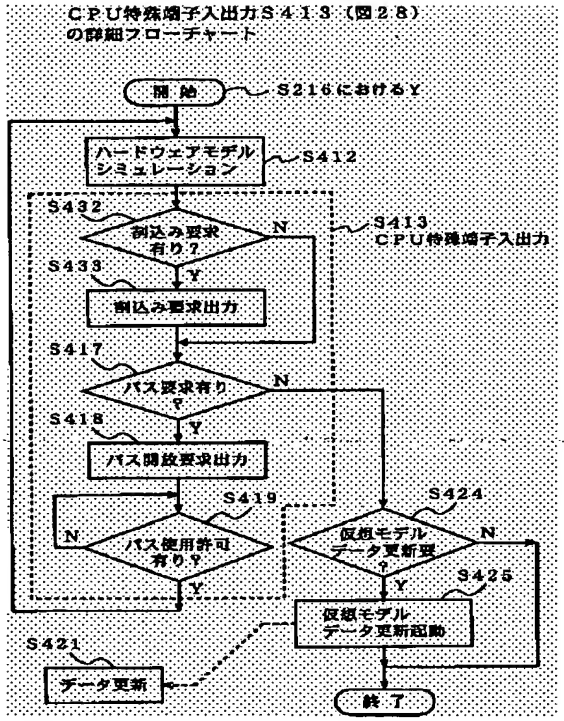
【図２７】


ＣＰＵモデルシミュレーションのフローチャート
開始
S361
割込み要求有？
S367
割込み受付可？
S362
バス解放要求有？
S368
割込み処理
S363
バス解放受付可？
S101
ＰＣ取得
S364
バス使用許可
S102
ＦＥＴＣＨ関数生成
S123
周辺回路処理サブルーチン
S365
バス解放
S104
命令読込
S105
命令解読
S366
バス解放？
S106
周辺回路終番？
S107
演算関数生成
S108
周辺回路処理サブルーチン
S109
命令実行
S110
ブレーク？
終了

【図２８】


周辺回路処理サブルーチンＳ１２３，Ｓ１２８（図２７）のフローチャート
開始
S431
仮想モデル呼出し？
S403
信号準備
S204
端子信号出力
S205
端子信号変化？
S206
端子信号入力
S207
バスＩ／Ｆ完了？
S208
S422
仮想モデルシミュレーション
データ更新
S421
S410
インターフェース関数Ｉ／Ｆ処理
S411
システムＩ／Ｆモデルシミュレーション
S212
端子信号出力
S213
端子信号変化？
S216
外部端子変化？
S214
端子信号入力
S215
システムＩ／Ｆ完了？
データ更新
S423
S412
ハードウェアモデルシミュレーション
S413
ＣＰＵ特殊端子入出力
S414
転送モデル登録？
S420
ハードウェアデータ更新処理
S425
仮想モデルデータ更新処理
戻る

【図２９】


ＣＰＵ特殊端子入出力Ｓ４１３（図２８）
の詳細フローチャート
開始
S216におけるY
ハードウェアモデル
シミュレーション
S412
S432
割込み要求
有り？
N
S413
ＣＰＵ特殊端子入出力
Y
S433
割込み要求出力
S417
バス要求有り
？
N
Y
S418
バス開放要求出力
S419
N
バス使用許可
有り？
Y
S424
仮想モデル
データ更新要
？
N
Y
S425
仮想モデル
データ更新起動
S421
データ更新
終了